INTERFACE™

MICROCOMPUTING FOR HOME AND SMALL BUSINESS VOL. 2. ISSUE 3. FEB. 1977 $1.50

Warp Factor

Microcomputer Stock Options

LEGION: Artificial Intelligence

Random Number Generator

8080 Memory Object Code Search Routine

Basic Algorithms For Math Functions

Meet the most powerful μC system available for dedicated work.

Yet it's only $595.*

***kit price**

Here's the muscle you've been telling us you wanted: a powerful Cromemco microcomputer in a style and price range ideal for your dedicated computer jobs—ideal for industrial, business, instrumentation and similar applications.

It's the new Cromemco Z-2 Computer System. Here's some of what you get in the Z-2 for only $595:

- The industry's fastest μP board (Cromemco's highly regarded 4 MHz, 250-nanosecond cycle time board).
- The power and convenience of the well-known Z-80 μP.
- A power supply you won't believe (+8V @ 30A, +18V and −18V @ 15A — ample power for additional peripherals such as floppy disk drives).
- A full-length *shielded* motherboard with 21 card slots.
- Power-on-jump circuitry to begin automatic program execution when power is turned on.
- S-100 bus.
- Standard rack-mount style construction.
- All-metal chassis and dust case.
- 110- or 220-volt operation.

DEDICATED APPLICATIONS

The new Z-2 is specifically designed as a powerful but economical dedicated computer for systems work. Notice that the front panel is entirely free of controls or switches of any kind. That makes the Z-2 virtually tamper-proof. No accidental program changes or surprise memory erasures.

FASTEST, MOST POWERFUL μC

Cromemco's microcomputers are the fastest and most powerful available. They use the Z-80 microprocessor which is widely regarded as the standard of the future. So you're in the technical fore with the Z-2.

BROAD SOFTWARE/PERIPHERALS SUPPORT

Since the Z-2 uses the Z-80, your present 8080 software can be used with the Z-2. Also, Cromemco offers broad software support including a monitor, assembler, and a BASIC interpreter.

The Z-2 uses the S-100 bus which is supported by the peripherals of dozens of manufacturers. Naturally, all Cromemco peripherals such as our 7-channel A/D and D/A converter, our well-known BYTESAVER with its built-in PROM programmer, our color graphics interface, etc., will also plug into the S-100 bus.

LOW, LOW PRICE

You'll be impressed with the Z-2's low price, technical excellence and quality. So see it right away at your computer store—or order directly from the factory.

Z-2 COMPUTER SYSTEM KIT (MODEL Z-2K) (includes 4 MHz μP card, full-length 21-card-slot motherboard, power supply, one card socket and card-guide set, and front panel; for rack mounting) . . . .$595.

Z-2 COMPUTER SYSTEM ASSEMBLED (MODEL Z-2W) (includes the above as well as all 21 sockets and card guides and a cooling fan; for rack mounting) . . . $995.

Shown with optional bench cabinet

Cromemco

incorporated

Specialists in computers and peripherals

2432 CHARLESTON RD., MOUNTAIN VIEW, CA 94043 • (415) 964-7400

CIRCLE INQUIRY NO. 6

INTERFACE AGE™

MICROCOMPUTING FOR HOME AND SMALL BUSINESS

VOL. 2 ISSUE 3 FEBRUARY 1977

FEATURES

SOFTWARE FEATURES

Cover Story

This month's cover is somewhat reminiscent of those first spectacular views of Mars as seen by researchers on earth as T.V. signals were beamed back over the many millions of miles through space.

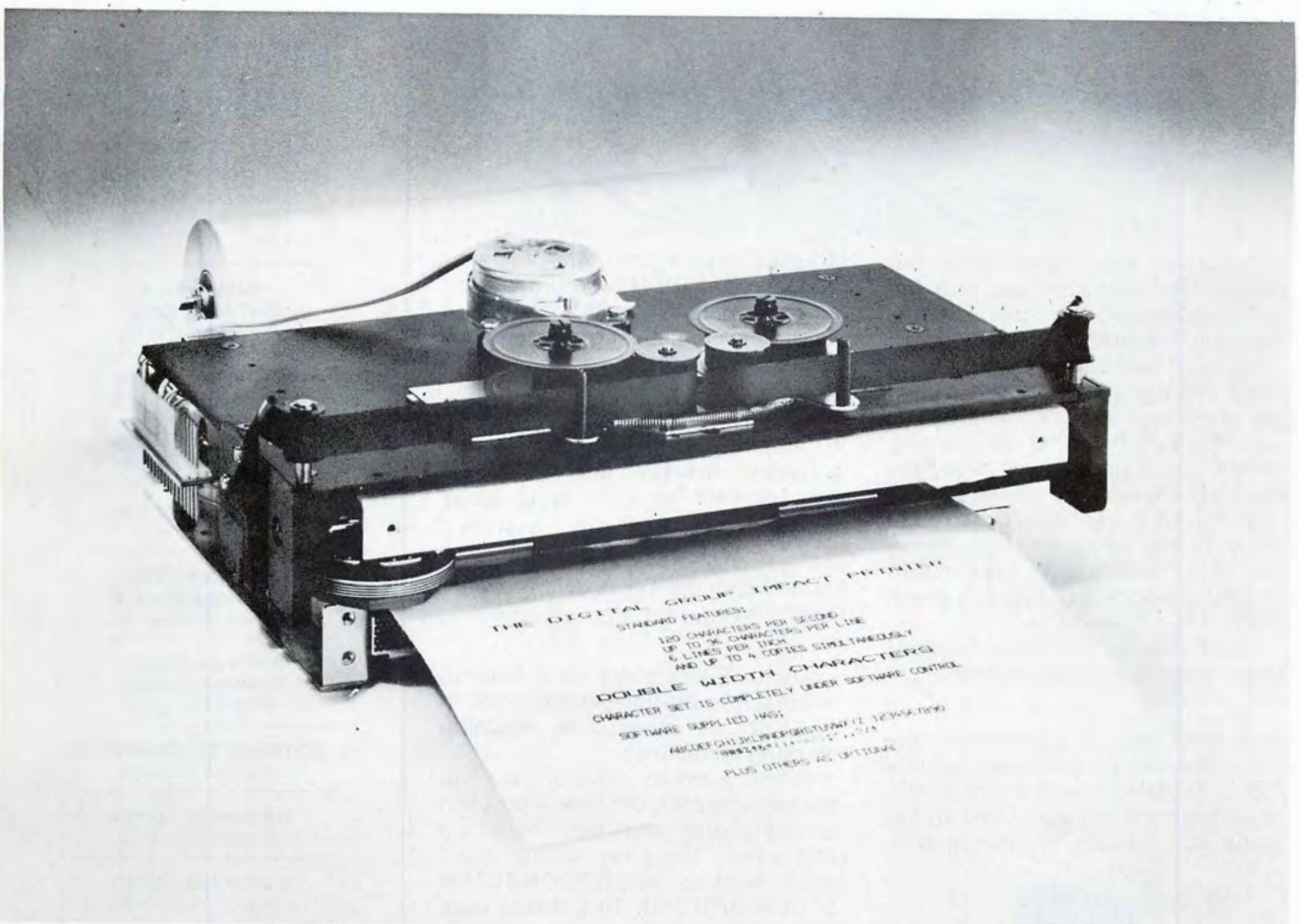
In reality it is a setting for a major announcement to take place in March. "Warp Factor" is a light hearted scenario highlighting the features of the newest microcomputer kit to enter the personal computing marketplace. Motorola is introducing the HEP EDUCATOR II, a small system designed for the economically oriented student or computer hobbyist. The total price for this nifty little box of computing tricks—$169.95.

You may start looking for Educator II in your local Byte Shop computer store or in many of the selected HEP distributors.

DEPARTMENTS

INTERFACE AGE Magazine, published monthly by McPheters, Wolfe & Jones, 13913 Artesia Blvd., Cerritos, Calif. 90701. Subscription rates: U.S. $10.00, Canada/Mexico $12.00, all other countries $18.00. Opinions expressed in by-lined articles do not necessarily reflect the opinion of this magazine or the publisher. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes an endorsement of the product or products by this magazine or the publisher.

POSTMASTER: Please send change of address form 3579 and undelivered copies to **INTERFACE AGE** Magazine, 13913 Artesia Blvd., Cerritos, Calif. 90701. Second-class postage paid at Artesia, California 90701 and at additional mailing offices.



Print Your Heart Out.

With help from the Digital Group, naturally.

Now, that small computer system you own or have been considering for personal or business use suddenly becomes a lot more usable—with the addition of a full-size *impact* printer from the Digital Group. A printer designed for small computers that need big output (like yours).

With the Digital Group printer, you can print your heart out...and it won't cost an arm and a leg. The Digital Group printer is available for less than $500. That's right—$500.

Just look at these specifications:

- Fast—120 characters per second
- 96 characters per line
- 12 characters per inch horizontal
- 6 lines per inch
- Makes up to 4 copies simultaneously
- Character set and pitch variable under software control—double width characters, etc.
- 5 x 7 character matrix
- Ribbon has built-in re-inkers for a life of 10,000,000 characters
- Paper can be either a standard 8½-inch roll, fanfold or cut page
- Interfaces to 8-bit parallel ports

There are lots of capabilities and outstanding features of the Digital Group printer...and (as always) the best news is our price. Kit prices start as low as $495 for the printer and interface card. It simply can't be beat.

Find out all the facts about the Digital Group printer now. Just fill in the coupon below or give us a call for the details. We think you'll find a place for our printer in your system ...and in your heart.

the digital group

P.O. Box 6528
Denver, Colorado 80206
(303) 777-7133

Quick. I want to print my heart out.
Send me all the details on your full-size impact printer.

Name ____________________

Address ____________________

City/State/Zip ____________________

Please print.

INTERFACIAL

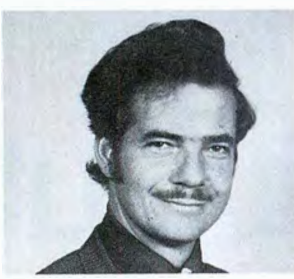
Departing this month from our normal immediate plunge into the goodies contained in this issue, I'm vectoring to a much requested priority.

Many of you have written asking "Who are the people whose names we see listed on the masthead?"

In this issue and for the coming several issues, we shall introduce the staff and include some of their background. It will help you to relate better to the magazine if you know who performs the tasks each month of getting a book together and eventually into your hands.

This month the Northwest Regional Editor and the Assistant Editor share the spotlight.

Dr. Adam Osborne is author of more than 50 works on computer technology and is familiar with a wide variety of assembly languages used by virtually all products on the market.



Formerly a computer systems engineer and process engineer with Shell Oil Corp., Dr. Osborne now heads his own firm, Osborne & Associates, Inc. He is a graduate of University of Birmingham, England, (B.Sc.), University of Delaware, (M.Ch.E. and Ph.D.)

Linda Folkard-Stengel has long experience in technical journalism, public relations and television journalism.

Quadrilingual, she was formerly with the Overseas Office of Design News and with the Mexico City-based official organ of the Association of Mexican Travel Agents, AMAV News. She served as TV News Film Librarian for Canadian Broadcasting Corporation's English-language anchor station CBLT, Toronto.

This month's issue has been artfully brushed with many valuable software articles answering the ever- present moan, "Now what do I do next with my system."

To help fill your system's endless appetite Tom Doyle presents an 8080 Octal Monitor program

Marlin Ellers gives us a fanciful approach to artificial intelligence in LEGION, an attempt at machine learning techniques.

If you want to make it "big" in the stock market, but have a problem understanding what you should do and when, then, my friend, don't miss reading MICROCOMPUTER STOCK OPTIONS. This theme puts your home computer to work profitably.

Everyone has wondered when Motorola would seriously get into hobby computing. Exclusive! The beginner now has the HEP EDUCATOR II, the first step to a bubble-packed computer kit available at over 1500 electronics stores across the nation. It is coming in March, but read about it now in WARP FACTORS by Tom Mazur.

Continuing with construction projects, Roger Brown offers another approach to A/D converters for your personal computing activities. Card-of-the-Month features the TDL-ZPU™ CARD with Roger Edelson casting a qualitative eye on this popular product.

For those who have the older teletypes, not useful to home computing applications, rejoice. A possible solution to a conversion for the rigs is outlined in COMPUTER COUPLING TO TELETYPE LOOPS by B.D. Lichtenwalner.

Understanding hardware and software concepts is outlined in ultra-simplistic terms by Ken Pugh in THE BLACK BOX.

Again Bob Stevens has corraled a herd of useful software items by many talented contributors in the avocational and vocational computing fields.

INTERFACE AGE MAGAZINE

13913 Artesia Boulevard
Cerritos, CA 90701
(213) 926-6629

PUBLISHER & EDITOR-IN-CHIEF
ROBERT S. JONES

ASSISTANT EDITOR
LINDA FOLKARD-STENGEL

ASSISTANT MANAGER
EVA YAKA

NORTHWESTERN REGIONAL EDITOR
ADAM OSBORNE PhD

NEW ENGLAND REGIONAL EDITOR
ROBERT E. TRIPP

EDITORIAL CO-ORDINATOR
WILLIAM SEVEDGE

HARDWARE EDITOR
ROGER EDELSON

SOFTWARE EDITOR
ROBERT STEVENS

ASSOCIATE EDITOR
JANE HILL ALDEN

PRODUCTION LAYOUT
MIKE ANTICH

ART
CHARLYNE LEVANDOWSKI

SUBSCRIPTION CIRCULATION
JO ANN FERGUSON

NATIONAL SALES MANAGER
BRUCE BERKEY

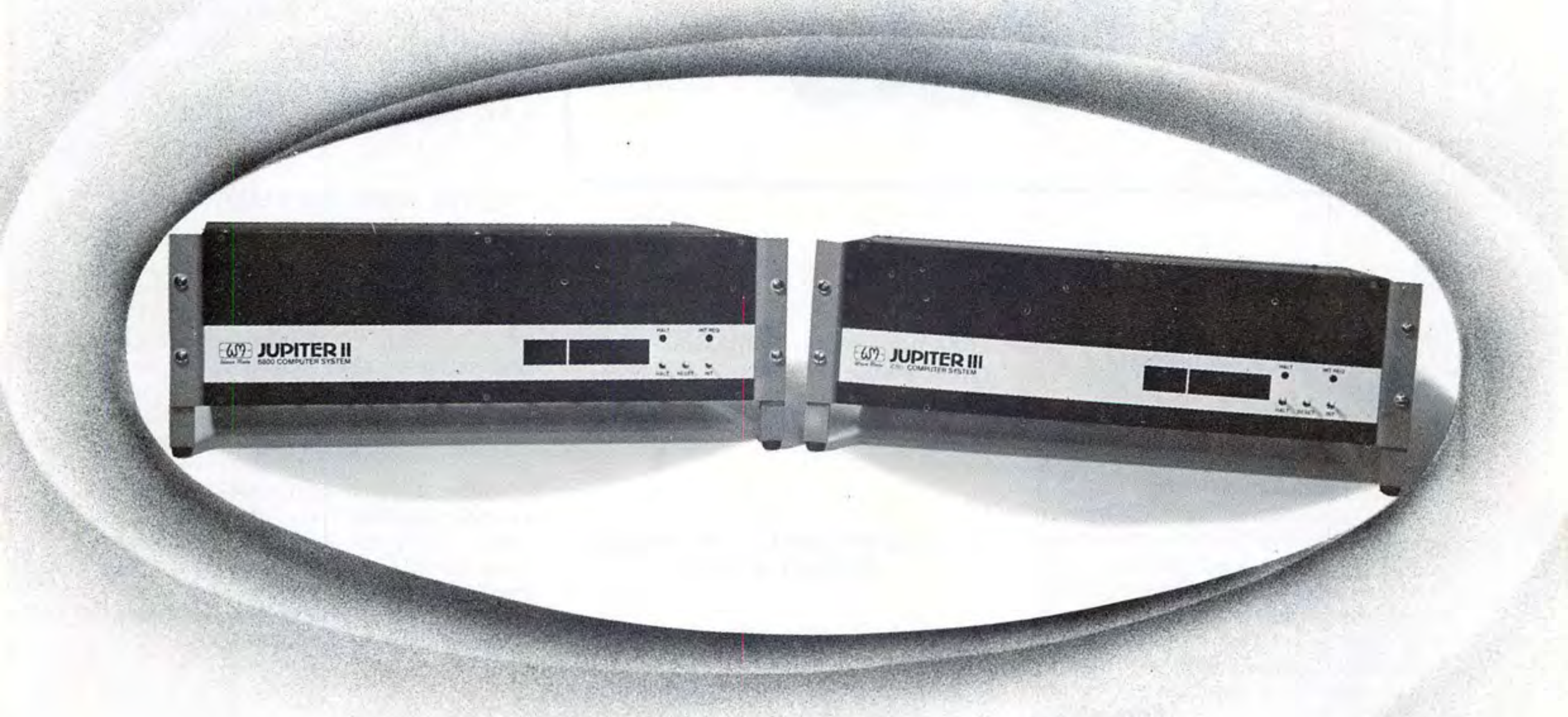
ADVERTISING—Representative
STEVE RICHARDS
ZACH BOVINETTE

RETAIL CIRCULATION
ZACH BOVINETTE

Editorial Correspondence
Direct all correspondence to the appropriate editor at: INTERFACE AGE magazine, P.O. Box 1234, Cerritos, CA 90701. Editorial contributions must be accompanied by return postage and will be handled with reasonable care, however, publisher assumes no responsibility for return or safety of manuscripts, art work, or models.

Advertising Inquiries
Direct all advertising inquiries to: Advertising Department, INTERFACE AGE magazine, 61 South Lake Avenue, P.O. Box 4566, Pasadena, CA 91106. (213) 795-7002.

POPULARITY EXPLOSION!



JUPITER II A
6800 System
$795

If you thought the quality of a wire-wrapp[illegible] system was beyond your price [illegible] Take a look at what we hav[illegible] now!

The Jupiter IIA and the Jupiter IIIA Basic computer systems. You get the system module cage with fully assembled backplane, fully assembled plug-in ferro-resonant power supply, front panel kit and your choice of 6800 or Z80 CPU module kit. All less than the price of the two best selling 8080 systems!

Plus you can choose from the fastest growing selection of memories and peripherals available from any manufacturer, like our 2KB EPROM/4KB RAM/serial RS-232 module which can transform your basic computer system into a real star.

And remember, all Wave Mate products meet the highest quality industrial standards, with rugged construction unmatched by anyone! Join the popularity explosion and get yours now! Write or call for more info and your closest Wave Mate authorized distributor.

You get your choice of microprocessors!
And you get wire-wrapped modules too!

Now you have a low cost way to get started into personal computing without sacrificing future growth capability!

Send information on: ☐ Jupiter IIA system
☐ Jupiter IIIA system

NAME ____________________

ADDRESS ____________________

CITY __________ STATE ______ ZIP ______

Wave Mate

WAVE MATE 1015 West 190th Street, Gardena, California 90248 Telephone (213) 329-8941
Dept 24

UPdate

NATIONAL SEMICONDUCTOR ANNOUNCES MICROPROCESSOR COURSES

Santa Clara, CA . . . National Semiconductor Corp. is offering a series of microprocessor training courses at its Santa Clara, California, facility, as well as SC/MP application courses in various western U.S. cities.

At National Semiconductor's Western Microprocessor Training Center, 1333 Lawrence Expressway, Santa Clara, CA, courses will be given in Microprocessor Fundamentals February 21-24, March 21-24, April 18-21, and June 6-9. Courses in Advanced Programming are scheduled for February 7-10 and May 9-12.

Courses on the application of National's PACE microprocessor will be offered at the Western Training Center on February 28 - March 3, April 25-28, and June 13-16. SC/MP application courses will be given March 7-10, March 28-31, May 2-5, and June 20-23.

In addition, courses on the applications of SC/MP (Simple, Cost-Effective Microprocessor) will be presented in Seattle, Washington, on February 14-18; in Edmonton, Canada, on February 21-25; in Denver, Colorado, March 14-18; and in Orange County, California, April 4-8. Class hours are 10 a.m. to 5 p.m.

The five-day courses given in the field include either a free SC/MP kit of a free SC/MP keyboard kit in the total fee of $395. The sessions discuss the fundamentals and tools of SC/MP microprocessors in detail to allow the student to solve any application problems he may encounter. They include complete SC/MP lab stations with one instructor for every six students for close lab supervision. In addition, the field courses feature hands-on time on two 16-bit IMP microprocessor systems (floppy disc) for high-speed software development and PROM programming.

Registration information may be obtained from local National Semiconductor sales offices or from National's Western Microprocessor Training Center in Santa Clara, California, (408) 247-7924.

AIAA SEMINAR ON PROPOSAL PREPARATION

A seminar in proposal preparation is to be held at Gene Autry Hotel, Palm Springs, California March 3-4, 1977 and at Hyatt/Union Square, San Francisco June 2-3, 1977. Sponsored by the American Institute of Aeronautics and Astronautics (AIAA), this two-day seminar deals exclusively with the methods for winning new business via the proposal route. The main themes propose to instruct how high technology products and services are procured by various U.S. government agencies and how the customer picks the winners. Presented over seventy times in three years in public and inhouse versions, the seminar has been consistently rated from excellent to outstanding by its over 3,000 attendees from some 200 companies, 80 government agencies and 50 universities.

The seminar is in nine parts under the following headings:
Part 1: The Ideal Proposal Organization - Authority and Responsibility; Part 2: The Important Pre-proposal Programs; Part 3: Initiation of the Proposal Effort; Part 4: How, When and Where to Spend Resources; Part 5: Learning to Communicate in a Written Proposal; Part 6: the Source Selection Process; Part 7: Actual Preparation of the Various Proposal Sections; Part 8: Post-proposal Activities and Tricks of the Trade. Registration information for individuals and groups may be obtained from AIAA Seminar, 444 W. Ocean Bl. P.O. Box 1710, Suite 1403, Long Beach, CA 90801 - 213 437-7465.

NEW CLUB

A new club is being formed for the exchange of information on use of Heathkit products by computer hobbyists. Write: Computer Heathkit Users' Group, c/o Charles Floto, 267 A Willow St., New Haven, CT 06511.

TRENTON COMPUTER FESTIVAL

The Second Trenton Computer Festival will be held at Trenton State College, Route 1, Trenton, N. J. from April 30, 1977 to May 1, 1977. Featured are a special conference sponsored by IEEE on consumer and hobby applications of microcomputers, exhibits, displays, contests, a huge flea market for hardware and software and manufacturers' booths.

General registration is $4.00 with a special students' rate of $2.00 and sales $2.00 per spot. For advanced registration or information call Jaci Di Paolo at (609)771-2487 or write Trenton Computer Festival, Trenton State College, Trenton N. J. 08625.

CALENDAR

Feb. 1 El Paso Computer Group meets at 7:00 p.m. for meetings place call (915) 544-1542.

Feb. 2 New England Computer Society, Inc. meets at 7:00 p.m. MITRE Corp., Cafeteria, Route 62 Bedford, MA. Call Dave Day at (603) 434-4239 for more information.

Feb. 2 Northwest Computer Club meets at 7:00 p.m., Pacific Science Center. Seattle, WA. Informal meetings. Details by calling (206) 524-6359.

Feb. 5 Louisville Area Computer Club meets at 1:00 p.m., Speed Auditorium, University of Louisville. General meetings, Contact Glen Darwin (502) 456-5589 or write 3028 Hunsinger Ln., Louisville, KY; 40200.

Feb. 6 Orange County Computer Club meets at 12:00 noon, California State University Fullerton, Administration Bldg., Room 321. For meeting agenda call: Lorin Mohler at (714) 998-5831.

Feb. 11 Crescent City Computer Club meets at 8:00 p.m., University of New Orleans, Lakefront Campus. For more information contact Bob Latham (504) 722-6321.

Feb. 11 Rochester Area Microcomputer Society meets at 6:30 p.m., Room 1030, Bldg. 9, Rochester, NY. Mailing address, RAMS, P.O. Box D, Rochester NY 14609.

Feb. 12 The Permian Basin Computer Group, Midland Chapter meets at 4:00 p.m., Midland College, Occupational Technology Bldg., Room 110, General Meeting. Call (915) 697-4697 for details.

Feb. 12 The Permain Basin Computer Group, Odessa Chapter meets at 1:00 p.m. Odessa College Electronics Technology Bldg., Room 203. Call (915) 332-9151 for more information.

Feb. 26 So. Cal. SWTPO MP 6800 Users Group meets at 10:00 a.m. at A-VID Electronics, 1655 E. 28th Street, Long Beach CA 70806.

Feb. 27 Chicago Area Computer Hobbyists Exchange (CACHE) meets at 12:00 noon, NIGAS Bldg., Schermer Rd., Glenview IL. Sick Computer Show. Bring in your problems and help repair them. Call Bill Precht at (312) 620-1671.

Feb. 28-Mar. 3 IEEE Computer Society's COMPCON '77 Spring beginning at 9:00 a.m. on Feb. 28 at the Jack Tar Hotel, San Francisco, California.

Mar. 2 New England Computer Society, Inc., meets at 7:00 p.m. MITRE Corp., Cafeteria, Route 62, Bedford, MA. General Meetings. Call Dave Day at (603) 434-4239.

Mar. 7 AMRAD Amateur Radio Research and Development Corp. meets at 8:00 p.m. in the Patrick Henry Library, Vienna, VA 22101. Call (703) 356-8918 for details.

March 9-11 DATACOMM '77 meets at Sheraton Park Hotel, Washington, DC. For more information contact Shaun Bresnahan, Director of Marketing, DATACOMM '77, 60 Austin St., Newtonville, MA 02160 or call (617) 964-4550.

Mar. 10 Rochester Area Microcomputer Society meets at 6:30 p.m. Room 1030, Bldg. 9, Rochester, NY. Mailing address, RAMS, P.O. Box D, Rochester NY 14609.

Mar. 22 So. Cal. SWTPO MP 6800 Users Group meets at 10:00 a.m. at A-VID Electronics, 1655 E. 28th Street, Long Beach, CA 70806.

Mar. 27 Chicago Area Computer Hobbyists Exchange (CACHE) meets at 12:00 noon, NIGAS Bldg., Schermer Rd., Glenview, IL. Small Business Opportunities Show. For further information contact Bill Precht at (312) 620-1671.

March 28-30 The 5th Annual Data Communications Conference and Exposition, INTERFACE '77 meets at Georgia World Congress Center, Atlanta, GA. For more information contact Bob Goolta, INTERFACE, 160 Speen Street, Framingham, MA 01701

April 19-21 Electro/77 meets at New York Coliseum and Hotel Americana, New York, New York. For more information write Elector/77, 999 N. Sepulveda Blvd., El Segundo, CA 90245 or call (800) 421-6816.

May 10-12 Chicagoland Business Services and Equipment Exposition meets at Expocenter/Chicago. For further information contact Carleton Rogers, Industrial and Scientific Conference Management Inc., 222 West Adams St., Chicago, IL 60606 or call (312) 263-4866.

June 13-16 NCC, Dallas, TX. For more information write AFIPS, 210 Summit Ave., Montvale, NJ 07645 or call (201) 391-9810.

COMING IN MARCH ISSUE...

New Products Guide

The Computer Even a Baby Can Use
by Kenneth Perry, Basil Steele Rocky Ridges and Harry Garland

An informative glance at the future of microprocessors in medical applications

Reflections on the Past and Thoughts About the Future of Semiconductor Technology
by Dr. C. Lester Hogan

Dr. Hogan has earned the title "Father of Semi-conductor Technology." In ***Reflections*** this eminent scientist relates an engrossing travelogue of the arduous adventures encounter by him and his colleagues throughout the years of development of the technology.

The Menace of the Micro World
Another essay and program on artificial intelligence

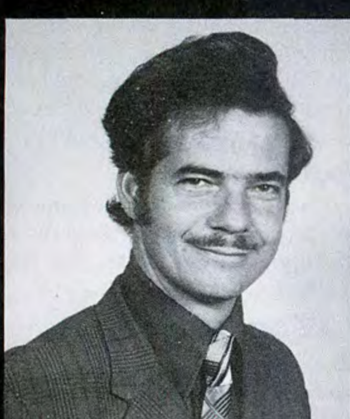
The Qube by Roger Garrett
Science-fiction has such a provocative way of becoming science-fact

Building a Digital Group System
by Donald O. Southwick
More valuable information for our hardware readers

And for our software readers, programs on
Depreciation Schedule Analysis
Checkbook Balance
Game of Life
—and others

. . . FROM THE FOUNTAINHEAD

By Adam Osborne



Let me introduce myself, I am Adam Osborne. Starting with this issue of INTERFACE AGE, I will be the Northwest regional editor. In every issue of INTERFACE AGE I will bring you recent microprocessor and microcomputer happenings in the Northwest. I will concentrate on developments of technical interests rather than reviewing society meetings. The companies that I will cover include all of those between San Jose and San Francisco — an area known affectionately as Silicon Valley. To me this is the "fountain head" from which microprocessors flow, hence the title I have chosen for my column. Microprocessor manufacturers within this small area include Intel, National Semiconductor, Fairchild, Intersil, Signetics, Scientific Micro Systems, Advanced Micro Devices, American Microsystems, Advanced Memory Systems, Electronic Arrays, Zilog and Synertek. Microcomputer manufacturers include IMSAI, Processor Technology, Cromemco, and Apple - and the Byte shops began here. These are the types of companies whose activities I will cover.

If you have information about your company about which you would like me to write, or if you have comments about what I have written, please call me directly at (415) 548-2805.

And now for some recent happenings of interest.

Guess who will sell the most 8-bit microprocessors in 1977? Intel with the 8080? Wrong! Fairchild with the F8 will outsell the 8080 at least 2-to-1 — providing Fairchild can manufacture enough product to fill orders. Yes, Fairchild is currently production bound. Fairchild has orders for between two million and three million F8 microprocessors in 1977.

But that is not very interesting to the low volume user because Fairchild's customer base consists of a few, very high volume accounts. Low volume and individual users don't take too readily to the F8.

Zilog is doing deceptively well. They are currently shipping approximately 5,000 CPUs a month. That does not sound like many when stacked up against 8080 or F8 sales, but numbers can be misleading. Big sales volume doesn't come until manufacturers put the Z80 into something that sells tens of thousands of units monthly. Typically it takes two years for a big user to bring a product with such large sales volume to the market. All those guys who will be buying 10,000 Z80 CPUs a month, now are buying 5 or 10 at a time. Zilog's present customer base suggests that Zilog will be selling many Z80's in 1979. Exxon be patient.

Intel's answer to the Z80, the 8085, is finally here. Is it a mouse or a powerhouse? It has the same instruction set as the 8080A, with just two additional I/O instructions — but it uses a single power supply and condenses the 8080A, 8224 and 8228 into a single chip. But the 8085 address and data buses are multiplexed. When Intel went from the 8008 to the 8080 I thought one of the big advances was to get rid of multiplex data and address buses.

The 8048 is Intel's answer to the F8. It is a single chip microcomputer, complete with CPU, interrupt logic, ROM, RAM, and I/O. The 8748 is a version with erasable ROM and that should make the chip the darling of all small manufacturers.

Fairchild cannot make erasable ROMs so they are coming up with a 40 pin package that combines the new, one chip 3859 F8 microcomputer with external erasable ROM. If the price is right, so is the product.

For all of you 8080A users who are into hardware, Advanced Micro Devices is coming out with a book which you must buy. It is packed with all kinds of good design tips. Contact Andrew Allison or Joe Kroeger for details. And plan on spending some money; the book costs $10 but it is worth the price.

Some big names are impressed enough with the microcomputer hobby market to jump on the band wagon. Pertec has bought MITS, the father of the Altair and the hobby market, and iCOM who built the first floppy disk drive to interface with microprocessors. iCOM uses Pertec floppy disk drives so Pertec is simply buying its best customer. But did you know that when the guys who started iCOM first went to Pertec to buy drives, they had to pay cash on the barrel head? Pertec thought they were too flaky for credit. Eighteen months later Pertec bought them for a quarter of a million Pertec shares. Not bad for 18 months' work.

But that is not the only Pertec story. Stu Mabon, Eric Dunstan, et. al started Pertec in a garage, back in 1968. Stu Mabon, Eric Dunstan, et. al. have quit Pertec and have started another new company. Wait for those guys to announce some big floppy disk breakthroughs in the next three months.

Have you heard the rumors about Heath Kit getting into the microcomputer hobby market? They are all true. Starting late this summer the Heath Company will be marketing a very well known, big name microcomputer system at a low, low price. And won't it be nice to have Heath Kit quality documentation to work from for a change?

How will all of the other retail outlets react to Heath Kit? Those with multiple lines will make it. Any retail store that markets just one microcomputer is severely handicapped in competition with retail outlets such as the Byte Shops which carry multiple lines. After all, what successful stereo and sound system stores try to sell just one line? I predict that the stores marketing just one microcomputer line will diversify or go under.

PRIME RADIX

ANNOUNCES THE 64K™

WE DO IT WITH MIRRORS!

(and some very sophisticated state-of-the-art memory design)

You've probably imagined that someday you'd like to own a computer system with a full complement of memory:

65,536 BYTES

Your dream can be a reality with the Prime Radix Corporation's 64K™ memory system at a very cost-effective price. And because it is a standalone memory system, you've got the advantage of greater flexibility not ordinarily available from add-in memory. Some of the features are:

- The 64K™ is fully buffered, presenting one TTL load to the memory bus.
- The 64K™ is digital group bus and ALTAIR™ bus compatible. **When ordering, you must specify the bus architecture.** A plugcard and cable will be furnished for the particular bus architecture you specify.
- The minimum complement of memory is 40K BYTES, with starting address locations at 0K, 8K, 16K, or 24K.
- The 64K™ comes assembled and tested with its own power supply, attractively housed in an aluminum cabinet, ready to plug into your system.
- Psuedo-static operation: on board refresh clock-generator provides processor independent refresh with no wait states. The 300NS worst case access time enhances high speed operation.
- Power/fail detection circuitry and battery backup will provide non-volatile memory (batteries are optional at extra cost).

LIST PRICE IS AS FOLLOWS:

40K	48K	56K	64K
$1490.00	$1580.00	$1670.00	$1750.00

We are offering a special introductory ten percent discount off list price on all orders received on or before February 28, 1977. Delivery will be made in the same sequence as orders are received. Please allow 3 to 6 weeks for delivery. Mastercharge and BankAmericard are accepted.

P.O. Box 11245 - Denver CO 80211 (303) 422-7040 or (303) 433-5630

Print Name

Address

City State Zip

☐ DIGITAL GROUP BUS

☐ ALTAIR™ BUS

☐ 64K @ $1750.00

☐ 56K @ $1670.00

☐ 48K @ $1580.00

☐ 40K @ $1490.00

☐ Check or M.O. enclosed

☐ Charge BAC

☐ Charge MC

(Please No C.O.D.'s or P.O.'s)

Make checks or money orders payable to:

PRIME RADIX, INC.
P.O. Box 11245
Denver, Colorado 80211

Credit Card Number

4 Numbers Above Name (MC) Good Thru

Signature

8080 OCTAL MONITOR PROGRAM

By Thomas E. Doyle

INTRODUCTION

This monitor program will enable you to control your 8080 system from an ASCII keyboard and a TTY or CRT readout. All standard front panel control functions (examine, examine next, deposit, load and run) are provided in octal format. Audio cassette input and output functions as well as a loader for MITS software are also included. Once you have this monitor in ROM, the drudgery of entering and reading data from the front panel switches and lights is all but eliminated.

PORT ASSIGNMENTS

The monitor is designed to operate in an 8080 system with keyboard data input on port 1 and keyboard data available checked on port 0, LSB (active low). Data output is also available on port 1 with terminal ready to receive data checked on port 0, MSB (active low). The audio cassette interface data are on port 7 with status checked on port 6. These standards correspond with MITS port assignments used for the ACR and serial I/O boards used in Basic and Package II software.

MEMORY REQUIREMENTS

Required are 512 bytes of memory which may be ROM or RAM. The program may be located anywhere in memory. A source object listing assembled to start at 376 000 is included at end of text. The best configuration is to put the monitor in EROM and locate it in a high memory location so it may reside concurrently with programs in the low RAM address. The program is organized as a series of general purpose subroutines which may be called from user programs.

MONITOR FUNCTION

The monitor functions are:

EXAMINE (E) — User types in octal address of memory location he wishes to examine and the computer prints out the address and data in octal format. HHH LLL : DDD :

EXAMINE NEXT — (SPACE) When in the *examine* mode the user may type the space bar and the computer will print the address and data for the next location in memory in octal format. HHH LLL : DDD :

DEPOSIT (D) — After examining a location the user may deposit new data in that location by typing the letter D followed by the new data in octal format. The computer checks for proper storage by typing out the octal equivalent of the data actually stored at that address. HHH LLL : DDD : D XXX XXX

Where XXX is the new octal data the user wishes to deposit at the address.

Note: You must examine a location before you can deposit data in that location.

RUN (R) — After examining a location the user may elect to start program execution at that address by typing the letter R.

Note: you must examine a location before you can begin program execution at that location.

LOAD (L) — After examining a location the user may elect to load octal data in sequential addresses by typing in L followed by the octal data. After the third digit in each octal number the computer will deposit the data in that address and check it as in the *deposit* mode, increment the address and automatically accept the next octal number. This mode is useful when you have a large amount of data to enter in sequential locations.

Note: You must examine the starting address before you can begin loading.

TAPE OUTPUT — Typing an O will select the tape output mode. The Computer will ask for the starting and ending addresses for the block of data you wish to put on cassette tape. After typing in the start and finish addresses, type space to begin output. The computer will record two STX characters (002) followed by the data. When it is finished the terminal will print: indicating it is through outputting data to the tape and is ready for a new command.

TAPE INPUT (I) — Typing an I will select the tape input mode. The computer will ask for the starting address where you wish to begin depositing the data from the cassette tape. Type a space following the address. When you are through entering the tape, type in a carriage return and the computer will print a : indicating it is ready for a new command.

Note: the system will not automatically return to command mode at the end of the tape. You must type carriage return.

BOOT STRAP (B) — Typing a B will copy a modified MITS cassette boot strap loader for 8K Basic down into RAM starting at location 000 000. After typing B, type a space and start your basic tape. No need to wait the 15 seconds. This feature will be greatly appreciated by those who have grown weary of toggle switching the boot strap in. Since the boot strap is copied into RAM you may make any necessary changes before starting execution.

Note: Typing a carriage return will return the monitor

to the command mode.

SUBROUTINES AVAILABLE FOR USER APPLICATIONS

Several of the subroutines used in the monitor may be used to handle I/O in user programs. These subroutines save all used registers so it is only necessary to call the subroutines.

PNT: Prints the contents of the accumulator on the terminal connected to port 1.

INP: Inputs data from the keyboard and returns with the data in the accumulator. The routine INP (page 2 000) is not used in the program. It is a general purpose routine for input from a keyboard and returning with the keyboard data in the accumulator. It was included as a general purpose routine for use in other programs.

CRL: Outputs an ASCII carriage return and linefeed.

SPC: Outputs an ASCII space.

POC: Prints the octal equivalent of the accumulator contents.

IOC: Inputs a 3 digit octal number from the keyboard and returns with data in accumulator.

TOT: Outputs the contents of the accumulator to the audio cassette interface.

TIN: Inputs from the audio cassette interface and returns with the data in the accumulator.

PROGRAM EXPANSION

Provision for simple expansion of the program is provided for by including a group of 3 NO-OPS in two critical locations. The end of the print (PNT) subroutine contains 3 NOP's which may be used for a call to a special I/O handler program (i.e. ASCII to BAUDOT converter). The input control (INC) subrouting inputs from the keyboard and runs through a series of comparisons to determine which command is present. If the program reaches the bottom of the list of comparisons without finding a match it enters a default routine which prints a ? indicating that an invalid command was present. Three NO - OPS are included just ahead of the default routine to allow calling another set of comparisons and associated jumps for additional commands.

This monitor is by no means the ultimate but it does provide all basic control of the microcomputer and I/O. The length was arbitrarily limited to 512 bytes so it could be held on two 1702 type PROMS. Possible areas for expansion are:

- Tape verify routine, after a block of memory has been recorded on audio cassette it could be read in and verified.
- HEX format, basic monitor functions handled in HEX format.
- Cassette I/O improvements, inclusion of file names and checksum on input and output routines.

NOTES ON MODIFIED MITS BOOTSTRAP LOADER

This routine copies the modified bootstrap loader, which is stored in the monitor program starting at (page 2 016), down to RAM starting at (000 000). After the routine has been copied down the routine waits for a key to be pressed on the keyboard. If any key other than a carriage return is pressed program execution will begin at the start of the bootstrap (000 024). THe loader that is copied down is for MITS 8K BASIC version 3.2.

If you wish to load software other than 8K BASIC, after typing B type a carriage return. You will now be back in the command mode and you can change what ever you need to by changing memory location (000 002) to 017 for 4-K Basic and Programming System II or to 057 for EXTENDED BASIC in the bootstrap. After making the changes, begin execution at (000 024).

The routine waits for the correct character marking the beginning of MITS tapes (Memory page 2 Address 044). For most current software this is 256. If you have an old version change location (000 027) to what ever character starts your tape. (Some older tapes use 175)

NOTES ON LISTING

The Program is contained on two 256 word pages. The first page contains the instructions for the commands. The second page contains the general purpose subroutines. The two pages do not have to be adjacent in memory. The listing includes object code for page 1 with a high address of 376 and page 2 with a high address of 377. These page references are underlined in the listing. Changing these page references in the jump and call commands will allow the program to run in any two blocks of memory. location of the stack

The first instruction (376 000) sets the stack pointer. Location of the stack pointer is dependent upon user's ram configuration and may be changed depending on your available memory.

PORT ASSIGNMENTS MAY BE CHANGED BY CHANGING

```
(PAGE 2 346)        FOR KEYBOARD STATUS
(PAGE 2 354)        FOR KEYBOARD DATA
(PAGE 2 364)        FOR DISPLAY STATUS
(PAGE 2 373)        FOR DISPLAY DATA
(PAGE 2 116) AND (PAGE 2 136) FOR ACR BOARD STATUS
(PAGE 2 125) AND (PAGE 2 146) FOR ACR BOARD DATA
```

PROGRAM EXECUTION BEGINS AT (PAGE 1 000)

COMMAND PROCESSING

```
MEMORY PAGE 1
000-061 377 037   INC: LXI SP      ;LOAD STACK POINTER
003-315 302 377   STA: CALL CLC    ;PRINT CR/LF AND :
006-315 345 377        CALL RCV    ;INPUT KEYBOARD DATA
011-376 105            CPI "E"     ;COMPARE FOR ASCII "E"
013-312 050 376        JZ EXA      ;JUMP TO EXAMINE ROUTINE IF "E"
016-376 111            CPI "I"     ;COMPARE ASCII "I"
020-312 202 376        JZ TIP      ;JUMP TO TAPE INPUT IF "I"
023-376 117            CPI "O"     ;COMPARE FOR ASCII "O"
025-312 246 376        JZ TOD      ;JUMP TO TAPE OUTPUT IF "O"
030-376 102            CPI"B"      ;COMPARE FOR ASCII "B"
032-312 345 376        JZ BSL      ;JUMP TO BOOT LOADER IF "B"
035-000                NOP         ;GROUP OF THREE NO OPS TO
036-000                NOP         ;ALLOW EXPANSION OF
037-000                NOP         ;COMMAND TABLE
040-076 077       DEF: MVI A, "?"  ;MOVE ASCII"?" TO A
042-315 362 377        CALL PNT    ;CALL PRINT SUBROUTINE
045-303 003 376        JMP STA     ;JUMP BACK TO START
```

EXAMINE

```
050-315 315 377   EXA: CALL CL>    ;PRINT CR/LF AND>
053-315 150 377        CALL LHK    ;LOAD H AND L FROM OCTAL INPUT
                                    FROM KEYBOARD
056-315 330 377   PXA: CALL CRL    ;PRINT CR/LF
061-315 166 377        CALL POH    ;PRINT OCTAL ADDRESS AND DATA
064-076 072            MVI A, ":"  ;MOVE ASCII ":" TO A
066-315 362 377        CALL PNT    ;CALL PRINT SUBROUTINE
071-315 345 377        CALL RCV    ;INPUT DATA FROM KEYBOARD
074-376 040            CPI " "     ;COMPARE ASCII "SPACE"
076-312 123 376        JZ EXN      ;JUMP TO EXAMINE NEXT IF
                                    "SPACE"
101-376 122            CPI "R"     ;COMPARE ASCII "R"
103-312 127 376        JZ RUN      ;JUMP TO RUN IF "R"
106-376 104            CPI "D"     ;COMPARE ASCII "D"
110-312 130 376        JZ DEP      ;JUMP TO DEPOSIT IF "D"
113-376 114            CPI "L"     ;COMPARE ASCII "L"
115-312 152 376        JZ LDE      ;JUMP TO LOAD IF "L"
120-303 040 376        JMP DEF     ;JUMP TO DEFAULT ROUTINE
```

EXAMINE NEXT

```
123-043           EXN: INX H       ;INCREMENT H AND L
124-303 056 376        JMP PXA     ;JUMP TO PRINT OCTAL ADDRESS
                                    AND DATA
```

RUN

```
127-351             RUN:PCHL          ;START EXECUTION AT ADDRESS
                                       REFERENCED BY H AND L
124-303 056 376         JMP PXA       ;JUMP TO PRINT OCTAL ADDRESS
                                       AND DATA
```

RUN

```
127-351             RUN:PCHL          ;START EXECUTION AT ADDRESS
                                       REFERENCED BY H AND L
```

DEPOSIT

```
130-315 272 377     DEP:CALL SPC      ;CALL PRINT SPACE SUBROUTINE
133-315 054 377         CALL OCI      ;CALL OCTAL DATA IN FROM
                                       KEYBOARD
136-167                 MOV M,A       ;STORE DATA IN MEMORY
137-315 272 377         CALL SPC      ;PRINT SPACE
142-176                 MOV A,M       ;MOVE DATA FROM MEMORY TO A
143-315 231 377         CALL POC      ;PRINT OCTAL EQUIVALENT OF
                                       DATA
146-043                 INX H         ;INCREMENT H AND L
147-303 056 376         JMP PXA       ;JUMP TO PRINT OCTAL ADDRESS
                                       AND DATA
```

LOAD

```
152-315 330 377     LDE: CALL CRL     ;PRINT CARRIAGE RETURN/LINE
                                       FEED
155-315 166 377         CALL POH      ;PRINT OCTAL EQUIVALENT OF
                                       ADDRESS AND DATA
160-315 272 377         CALL SPC      ;PRINT ASCII "SPACE"
163-315 054 377         CALL OCI      ;LOAD OCTAL DATA FROM
                                       KEYBOARD
166-167                 MOV M. A      ;MOVE DATA TO MEMORY
167-315 272 377         CALL SPC      ;PRINT ASCII "SPACE"
172-176                 MOV A. M      ;MOVE DATA FROM MEMORY
173-315 231 377         CALL POC      ;PRINT OCTAL EQUIVALENT OF
                                       DATA
176-043                 INX H         ;INCREMENT H AND L
177-303 152 376         JMP LDE       ;JUMP FOR NEXT BYTE
```

TAPE IN

```
202-315 315 377     TIP: CALL CL>     ;PRINT CR/LF AND>
205-315 150 377         CALL LHK      ;LOAD H AND L FROM KEYBOARD
210-315 302 377         CALL CLC      ;PRINT CR/LF AND :
213-315 345 377         CALL RCV      ;WAIT FOR A KEY ON KEYBOARD TO
                                       BE DEPRESSED
216-315 127 377     TSC:CALL TIN      ;INPUT DATA FROM ACR BOARD
221-376 002             CPI "2"       ;CHECK FOR STX (002)
223-302 216 376         JNZ TSC       ;JUMP IF DATA IS NOT STX
226-315 127 377     TSD:CALL TIN      ;INPUT DATA FROM ACR BOARD
231-376 002             CPI "2"       ;CHECK FOR STX (002)
233-302 226 376         JNZ TSD       ;JUMP IF DATA IS NOT STX
236-315 127 377     TSN: CALL TIN     ;INPUT DATA FROM ACR BOARD
241-167                 MOV M. A      ;STORE DATA
242-043                 INX H. L      ;INCREMENT H AND L
243-303 236 376         JMP TSN       ;JUMP FOR NEXT BYTE
```

TAPE OUT

```
246-315 315 377     TOD:CALL CL>      ;PRINT CR/LF AND>
251-315 150 377         CALL LHK      ;LOAD H AND L FROM KEYBOARD
254-315 272 377         CALL SPC      ;PRINT SPACE
257-076 124             MVIA. "T"     ;MOVE ASCII T TO ACCUMULATOR
261-315 362 377         CALL PNT      ;PRINT T
264-076 117             MVI A. "O"    ;MOVE ASCII O TO ACCUMULATOR
266-315 362 377         CALL PNT      ;PRINT O
271-315 272 377         CALL SPC      ;PRINT A SPACE
274-345                 PUSH H        ;PUSH H AND L
275-315 150 377         CALL LHK      ;LOAD H AND L FROM KEYBOARD
300-124                 MOV D. H      ;MOVE H TO D
301-135                 MOV E. L      ;MOVE L TO E
302-341                 POP H         ;POP H AND L
303-315 330 377         CALL CRL      ;PRINT CR/LF
306-076 002             MVI A. "2"    ;MOVE STX "002" TO
                                       ACCUMULATOR
310-315113 377          CALL TOT      ;RECORD STX ON TAPE
313-076 002             MVI A. "2"    ;MOVE STX "002" TO
                                       ACCUMULATOR
315-315 113 377         CALL TOT      ;RECORD STX ON TAPE
320-176             TOE:MOV A. M      ;MOVE DATA FROM MEMORY TO
                                       ACCUMULATOR
321-315 113 377         CALL TOT      ;RECORD DATA ON TAPE
324-174                 MOV A. H      ;MOVE H TO A
325-272                 CMP D         ;COMPARE D WITH H
326-302 341 376         JNZ TON       ;JUMP IF D NOT = H
331-175                 MOV A. L      ;MOVE L TO A
332-273                 CMP E         ;COMPARE E WITH L
333-302 341 376         JNZ TON       ;JUMP IF E NOT = L
336-303 003 376         JMP STA       ;JUMP BACK TO MONITOR SINCE
                                       ENTIRE BLOCK HAS BEEN
                                       RECORDED
341-043             TON:INX H         ;INCREMENT H AND L
342-303 320 376         JMP TOE       ;JUMP FOR NEXT BYTE
```

MITS BOOTSTRAP

```
345-021 000 000     BSL: LXI D. "0. 0"     ;LOAD D AND E WITH 000 000
350-041 016 377         LXI H. "377 016"   ;LOAD H AND L WITH 376 016
353-176                 MOVA. M       ;MOVE M TO A
354-353             BSN:XCHG          ;EXCHANGE H AND L WITH D AND E
355-167                 MOV M. A      ;STORE DATA
356-353                 XCHG          ;EXCHANGE H AND L WITH D AND E
357-175                 MOV A. L      ;MOVE L TO A
360-376 055             CPI "055"     ;CHECK FOR END
362-312 372 376         JZ END        ;JUMP IF END
365-043                 INX H         ;INCREMENT H AND L
366-023                 INX D         ;INCREMENT D AND E
367-303 353 376         JMP BSN       ;JUMP FOR NEXT BYTE
372-315 345 377     END:CALL RCV      ;WAIT FOR KEY ON KEYBOARD TO
                                       BE DEPRESSED
375-303 024 000         JMP "000 024" ;JUMP TO 000 024 WHICH IS START
                                       OF BOOTSTRAP LOADER
                                       PROGRAM
000-333 000         INP: INO          ;INPUT KEYBOARD STATUS
002-017                 RRC           ;ROTATE RIGHT
003-332 000 377         JC INP        ;JUMP BACK IF NO DATA
                                       AVAILABLE
006-333 001             IN 1          ;INPUT KEYBOARD DATA
010-000                 NOP           ;NO OPERATION
011-000                 NOP           ;NO OPERATION
012-000                 NOP           ;NO OPERATION
013-303 362 377         JMP PNT       ;JUMP TO PRINT SUBROUTINE
016-041 256 037     BSP LXI H         ;LOAD H AND L WITH (037 256)
021-061 022 000         LXI SP        ;LOAD STACK POINTER WITH (000
                                       022)
024-333 006             IN 6          ;INPUT ACR STATUS
026-017                 RRC           ;ROTATE RIGHT
027-330                 RC            ;RETURN IF CARRY
030-333 007             IN 7          ;INPUT ACR DATA
032-275                 CMP L         ;COMPARE L
033-310                 RZ            ;RETURN IF ZERO
034-055                 DCR L         ;DECREMENT L
035-167                 MOV M. A      ;MOVE DATA TO MEMORY
036-300                 RNZ           ;RETURN IF NOT ZERO
037-351                 PCH L         ;EXCHANGE PC WITH H AND L
040-003                 INX B         ;INCREMENT B AND C
041-000                 NOP           ;NO OPERATION
042-333 007             IN 7          ;INPUT ACR DATA
044-376 256             CPI 256       ;COMPARE FOR CHARACTER
                                       MARKING 256
046-302 024 000         JNZ (000 024) ;JUMP BACK IF DATA IS NOT 256
051-303 000 000         JMP (000 000) ;JUMP TO START OF BOOTSTRAP
054-305             OCI: PUSH B       ;PUSH B
055-006 000             MVI B         ;MOVE 000 TO B
057-315 345 377         CALL RCV      ;CALL KEYBOARD DATA INPUT
062-346 003             ANI "3"       ;AND IMMEDIATE (MASK 2 LSB'S)
```

Now you can buy an Altair™ 8800b or an Altair 680b computer right off the shelf. Altair plug-in boards, peripherals, software and manuals are also available. Check the list below for the MITS dealer in your area.

off the shelf.

RETAIL COMPUTER STORE, INC.
Tim & Susanne Broom
410 NE 72nd St.
SEATTLE, WA 98115
(206) 524-4101

COMPUTER KITS (S. F. area)
Pete Roberts
1044 University Ave.
BERKELEY, CA 94710
(415) 845-5300

THE COMPUTER STORE
(Arrowhead Computer Co.)
Dick Heiser
820 Broadway
SANTA MONICA, CA 90401
(213) 451-0713

GATEWAY ELECTRONICS, INC. OF COLORADO
George Mensik
2839 W. 44th Ave.
DENVER, CO 80211
(303) 458-5444

COMPUTER SHACK
Pete Conner
3120 San Mateo NE
ALBUQUERQUE, NM 87110
(505) 883-8282, 883-8283

GLOBAL ENGINEERING CO.
5416 South Yale
TULSA, OKLA. 74145
(918) 452-2567

COMPUTER PRODUCTS UNLIMITED
Harry & Margaret Mohrmann
4216 West 12th
LITTLE ROCK, AR 72204
(501) 666-2839

GATEWAY ELECTRONICS, INC.
Harry & Margaret Mohrmann
Lou Elkins, Stuart Bartfield
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314) 427-6116

CHICAGO COMPUTER STORE
Lou Van Eperen
517 Talcott Rd.
PARK RIDGE, IL 60068
(312) 823-2388

THE COMPUTER ROOM
3938 Beau D'Rue Drive
Eagan, MN 55122
Dale Hagert, Bob Raemer
(612) 452-2567

BYTE'TRONICS
John & Stan Morrow
Suite 103
1600 Hayes St.
NASHVILLE, TN 37203
(615) 329-1979

THE COMPUTER SYSTEMCENTER
Jim Dunion, Rich Stafford,
Steven Mann, Ron Roberts
3330 Piedmont Road
ATLANTA, GA 30305
(404) 231-1691

THE COMPUTER STORE, INC.
Sid Halligan
120 Cambridge St.
BURLINGTON, MA 01803
(617) 272-8770
Jeff Feldman, Service Dept.

THE COMPUTER STORE OF NEW YORK
Bob Arning
55 West 39th St.
NEW YORK, NEW YORK 10018
(212) 221-1404

THE COMPUTER STORE OF ANN ARBOR
Peter Blond
310 East Washington Street
ANN ARBOR, MI 48104
(313) 995-7616

THE COMPUTER STORE, INC. (Hartford area)
George & Susan Gilpatrick
63 South Main Street
WINDSOR LOCKS, CT 06096
(203) 627-0188

MICROSYSTEMS (Washington, D.C.)
Gloria & Russell Banks
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703) 569-1110

THE COMPUTER STORE
Stephen Payne
1114 Charleston National Plaza
CHARLESTON, W. VA. 25301
(304) 343-4607

MARSH DATA SYSTEMS
Don Marsh
5405 B Southern Comfort Blvd.
TAMPA, FL 33614
(813) 886-9890

mits

CIRCLE INQUIRY NO. 15

```
064-037              RAR          ;ROTATE RIGHT
065-037              RAR          ;THREE TIMES
066-037              RAR          ;
067-200              ADD B        ;ADD B
070-107              MOV B.A      ;MOVE A TO B
071-315 345 377      CALL RCV     ;CALL KEYBOARD DATA INPUT
074-346 007          ANI"7"       ;AND IMMEDIATE (MASK 3 LSB'S)
076-007              RLC          ;ROTATE LEFT THROUGH
077-007              RLC          ;CARRY THREE
100-007              RLC          ;TIMES
101-200              ADD B        ;ADD B
102-107              MOV B.A      ;MOVE A TO B
103-315 345 377      CALL RCV     ;CALL KEYBOARD DATA INPUT
106-346 007          ANI "7"      ;AND IMMEDIATE (MASK 3 LSB'S)
110-200              AOO B        ;ADD B
111-301              POP B        ;POP B AND C REGISTERS
112-311              RET          ;UNCONDITIONAL RETURN
113-365         TOT: PUSH PSW     ;PUSH ACCUMULATOR
114-067              STC          ;SET CARRY
115-333 006     TOI: IN 6         ;INPUT ACR BOARD STATUS
117-007              RLC          ;ROTATE LEFT THROUGH CARRY
120-332 115 377      JC TOI       ;JUMP IF VART BUFFER FULL
123-361              POP PSW      ;POP ACCUMULATOR
124-323 007          OUT 7        ;OUTPUT DATA TO ACR BOARD
126-311              RET          ;UNCONDITIONAL RETURN
127-333 001     TIN: IN 1         ;INPUT KEYBOARD DATA
131-376 015          CPI "CR"     ;COMPARE FOR ASCII CARRIAGE
                                   RETURN
133-312 003 376      JZ STA       ;JUMP IF ASCII CARRIAGE RETURN
136-067              STC          ;SET CARRY BIT
137-333 006          IN 6         ;INPUT ACR BOARD STATUS
141-017              RRC          ;ROTATE RIGHT THROUGH CARRY
142-332 127 377      JC TIN       ;JUMP IF NO DATA AVAILABLE
145-333 007          IN 7         ;INPUT ACR BOARD DATA
147-311              RET          ;UNCONDITIONAL RETURN
150-365          LHK:PUSH PSW     ;PUSH ACCUMULATOR
151-315 054 377      CALL OCI     ;CALL OCTAL IN
154-147              MOV H.A      ;MOVE ACCUMULATOR TO H
155-315 272 377      CALL SPC     ;PRINT A SPACE
160-315 054 377      CALL OCI     ;CALL OCTAL INPUT
163-157              MOV L.A      ;MOVE ACCUMULATOR TO L
164-361              POP PSW      ;POP ACCUMULATOR
165-311              RET          ;UNCONDITIONAL RETURN
166-365          POH:PUSH PSW     ;PUSH ACCUMULATOR
167-315 204 377      CALL POA     ;PRINT OCTAL EQUIVALENT OF
                                   ADDRESS
172-076 072          MVIA. ":"    ;MOVE ASCII : TO ACCUMULATOR
174-315 362 377      CALL PNT     ;PRINT :
177-315 222 377      CALL POD     ;PRINT OCTAL EQUIVALENT OF
                                   DATA
202-361              POP PSW      ;POP ACCUMULATOR
203-311              RET          ;UNCONDITIONAL RETURN
204-365          POA:PUSH PSW     ;PUSH ACCUMULATOR
205-174              MOV A.H      ;MOVE H REGISTER TO
                                   ACCUMULATOR
206-315 231 377      CALL POC     ;PRINT OCTAL EQUIVALENT OF H
                                   REGISTER
211-315 272 377      CALL SPC     ;PRINT A SPACE
214-175              MOV A. L     ;MOVE L REGISTER TO
                                   ACCUMULATOR
215-315 231 377      CALL POC     ;PRINT OCTAL EQUIVALENT OF L
                                   REGISTER
220-361              POP PSW      ;POP ACCUMULATOR
221-311              RET          ;UNCONDITIONAL RETURN
222-365          POD:PUSH PSW     ;PUSH ACCUMULATOR
223-176              MOV A.M      ;MOVE MEMORY DATA TO
                                   ACCUMULATOR
224-315 231 377      CALL POC     ;PRINT ACCUMULATOR OCTAL
                                   EQUIVALENT
227-361              POP PSW      ;POP ACCUMULATOR
230-311              RET          ;UNCONDITIONAL RETURN
231-345          POC:PUSH H       ;PUSH H AND L REGISTERS
232-157              MOV L.A      ;MOVE ACCUMULATOR TO L
233-007              RLC          ;ROTATE LEFT TWICE
234-007              RLC          ;
235-346 003          ANI "3"      ;MASK OFF ALL BUT 3 BITS
237-366 260          ORI 260      ;FORM ASCII DIGIT
241-315 362 377      CALL PNT     ;PRINT FIRST OCTAL DIGIT
244-175              MOV A.L      ;MOVE L TO ACCUMULATOR
245-017              RRC          ;ROTATE RIGHT 3 TIMES
246-017              RRC          ;
247-017              RRC          ;
250-346 007          ANI "7"      ;MASK OFF ALL BUT 3 BITS
252-366 260          ORI 260      ;FORM ASCII DIGIT
254-315 362 377      CALL PNT     ;PRINT SECOND OCTAL DIGIT
257-175              MOV A. L     ;MOVE L TO ACCUMULATOR
260-346 007          ANI "7"      ;MASK OFF ALL BUT 3 BITS
262-366 260          ORI 260      ;FORM ASCII DIGIT
264-315 362 377      CALL PNT     ;PRINT THIRD OCTAL DIGIT
267-175              MOV A.L      ;MOVE L TO ACCUMULATOR
270-341              POP H        ;POP H AND L REGISTERS
271-311              RET          ;UNCONDITIONAL RETURN
272-365         SPC: PUSH PSW     ;PUSH ACCUMULATOR
273-076 040          MVI A " "    ;MOVE ASCII SPACE TO
                                   ACCUMULATOR
275-315 362 377      CALL PNT     ;PRINT SPACE
300-361              POP PSW      ;POP PSW
301-311              RET          ;UNCONDITIONAL RETURN
302-365         CLC: PUSH PSW     ;PUSH ACCUMULATOR
303-315 330 377      CALL CRL     ;PRINT CARRIAGE RETURN AND
                                   LINE FEED
306-076 072          MVI A. ":"   ;MOVE ASCII : TO ACCUMULATOR
310-315 362 377      CALL PNT     ;PRINT :
313-361              POP PSW      ;POP ACCUMULATOR
314-311              RET          ;UNCONDITIONAL RETURN
315-365          CL>:PUSH PSW     ;PUSH ACCUMULATOR
316-315 330 377      CALL CRL     ;PRINT CARRIAGE RETURN AND
                                   LINE FEED
321-076 076          MVIA. ">"    ;MOVE ASCII> TO ACCUMULATOR
323-315 362 377      CALL PNT     ;PRINT
326-361              POP PSW      ;POP ACCUMULATOR
327-311              RET          ;UNCONDITIONAL RETURN
330-365         CRL: PUSH PSW     ;PUSH ACCUMULATOR
331-076 015          MVI A. "CR"  ;MOVE ASCII CARRIAGE RETURN
                                   TO ACCUMULATOR
333-315 362 377      CALL PNT     ;PRINT CARRIAGE RETURN
336-076 012          MVIA. "LF"   ;MOVE ASCII LINE FEED TO ACCUM
340-315 362 377      CALL PNT     ;PRINT LINE FEED
343-361              POP PSW      ;POP ACCUMULATOR
344-311              RET          ;UNCONDITIONAL RETURN
345-333 000      RCV:IN 0         ;INPUT STATUS CHANNEL
347-017              RRC          ;CHECK LSB
350-332 345 377      JC RCV       ;JUMP BACK IF NO KEYBOARD
                                   DATA
353-333 001          IN 1         ;INPUT KEYBOARD DATA
355-376 015          CPI "CR"     ;COMPARE FOR ASCII CARRIAGE
                                   RETURN
357-312 003 376      JZ "STA"     ;JUMP TO START IF CARRIAGE
                                   RETURN
362-365         PNT: PUSH PSW     ;PUSH ACCUMULATOR
363-333 000      PNA:IN 0         ;INPUT STATUS CHANNEL
365-007              RLC          ;CHECK MSB
366-332 363 377      JC PNA       ;JUMP BACK IF TERMINAL NOT
                                   READY
371-361              POP PSW      ;POP ACCUMULATOR
372-323 001          OUT 1        ;PRINT ACCUMULATOR CONTENTS
374-000              NOP          ;NO-OP'S TO ALLOW CALL TO
375-000              NOP          ;SPECIAL I/O HANDLER
376-000              NOP          ;ROUTINE
377-311              RET          ;UNCONDITIONAL RETURN
```

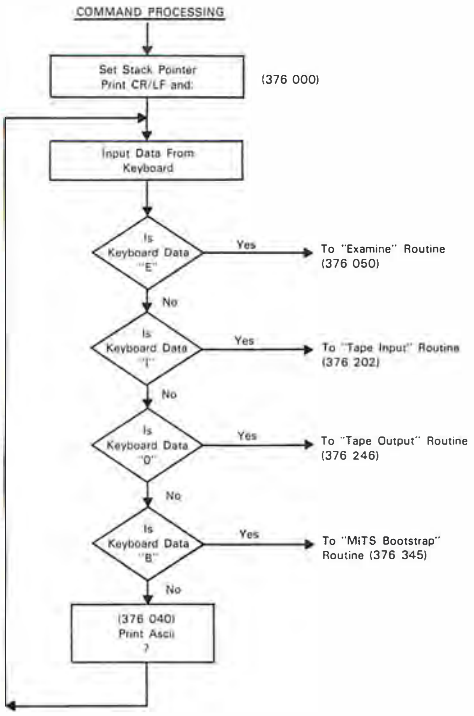


Figure 1.

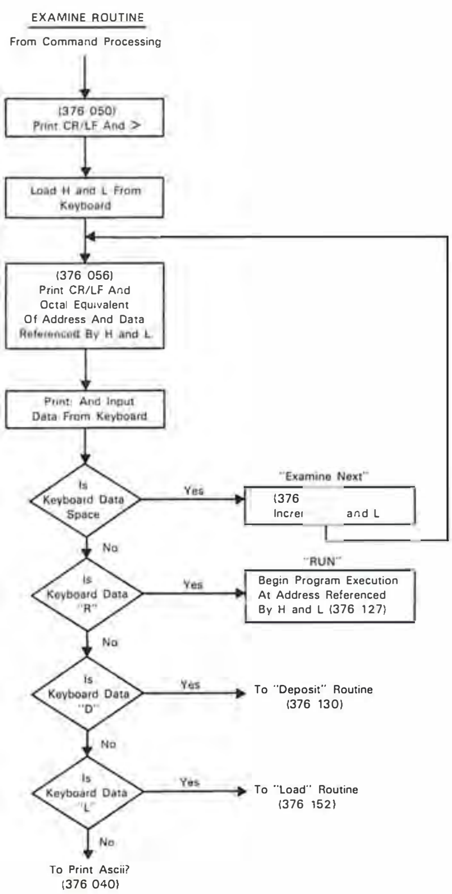


Figure 2

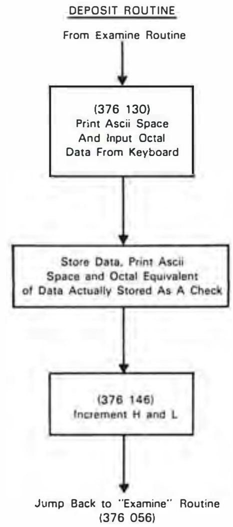
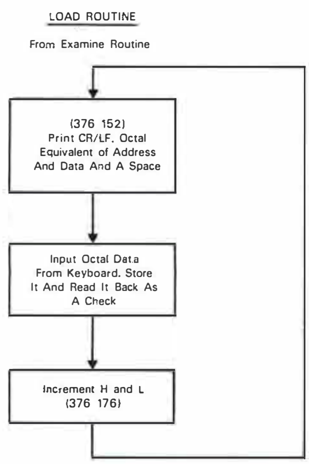
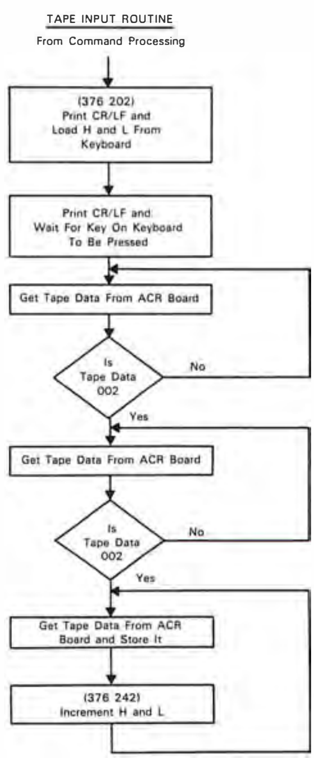


Figure 3.



NOTE: Typing a carraige return instead of octal data will cause a return to the command process routine.

Figure 4.



NOTE: After the tape has been read in, type a carriage return to return to the command process routine.

Figure 5.

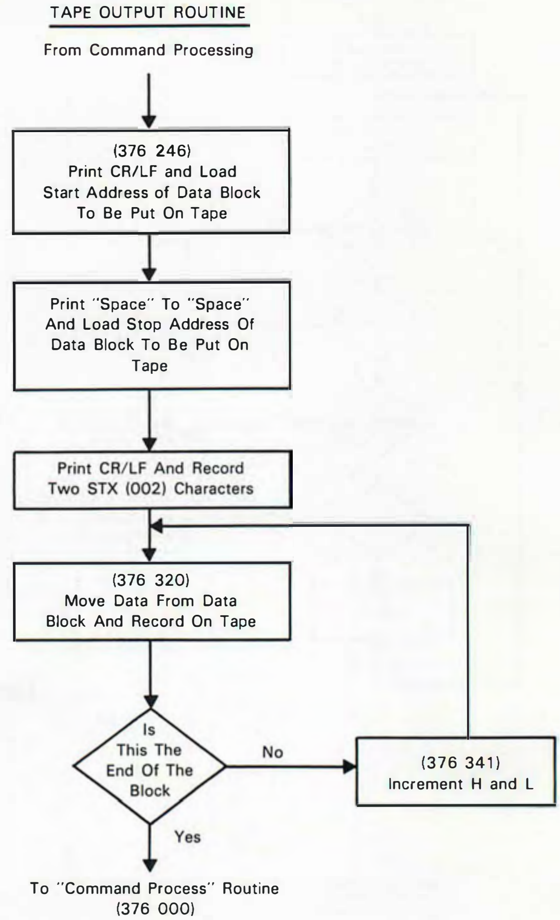


Figure 6.

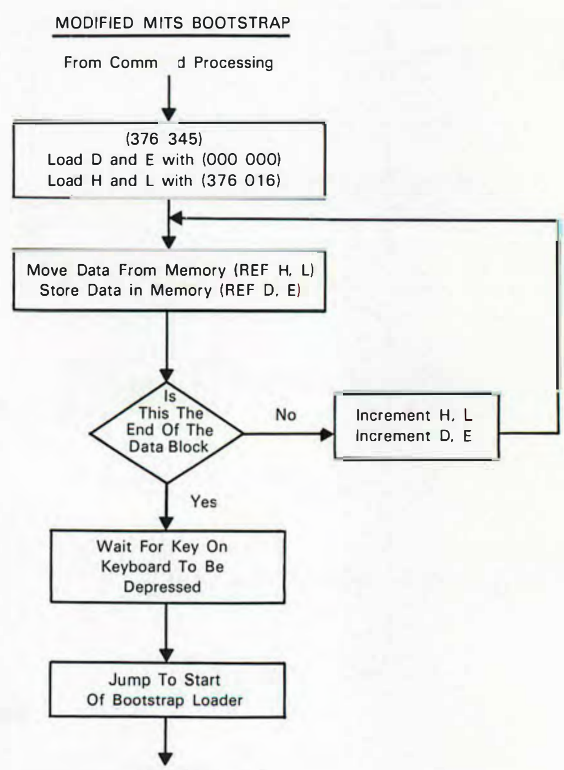
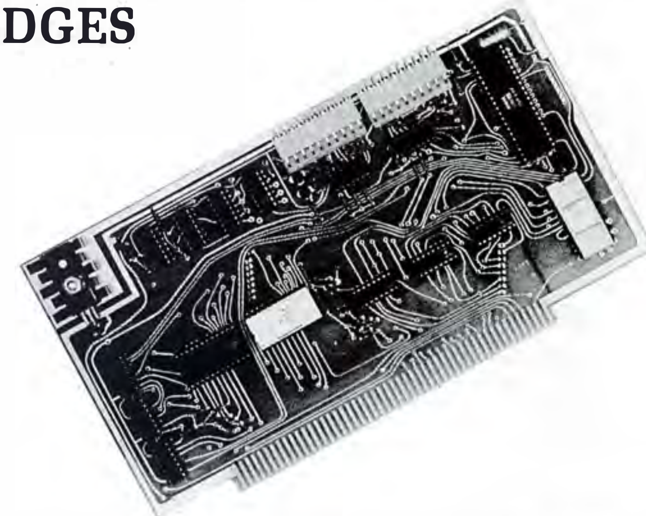


Figure 7.

DIGITAL DATA RECORDERS

USING 3M DATA CARTRIDGES



MODEL 3M3 $199.95

(Price Increases to $220.00
effective 1 April 77)

2SIO(R) CONTROLLER $190.00

MODEL 3M3 Featuring the radically new "Uniboard" method of construction for data cartridge drives. The major computer makers are changing to cartridges at a rapid pace because of the freedom from binding and greater data reliability. Operates in the phase encoded self-clocking mode which provides greatly enhanced freedom from speed variation problems and allows 100% tape interchangeability between units.

Uses the 3M Data Cartridge, model DC 300. This cartridge contains 300 feet of .250 tape in a sealed plastic container. Using four tracks you can record nearly 2 megabytes of data on a cartridge.

Specifications: Full software control of record, play, fast forward and rewind. LED indicates inter-record gaps. EOT and BOT are sensed and automatically shut down recorder. Feedback signals send reset and inter-record gap signals back to the computer so that software searching for inter-record gaps at high speed can be accomplished. Can also be operated manually by means of the switches on top which parallel the software control signals. $199.95 until April 1, 77. $220.00 after April 1, 77. Includes Phase Encoder Board (ACI).

For 8080, 8085, and Z-80 users:

Comes complete with software listing for the programs in the 2SIO(R) ROMs. Can be controlled by any of the commonly used I/O boards. Send for complete documentation and interfacing instru 3M3 and 2SIO(R) ($3.00). These programs provide full software control.

2SIO(R) CONTROLLER (Bootstrap Eliminator)

This is a complete 8080, 8085, or Z80 system controller. It provides the terminal I/O (RS232, 20 ma., or TTL) and the data cartridge I/O, plus the motor controlling parallel I/O latches. One kilobyte of on board ROM provides turn on and go control of your Altair or Imsai. No more bootstrapping. Loads and Dumps memory in Hex on the terminal, formats tape cartridge files, has word processing and paper tape routines. Best of all, it has the search routines to locate files and records by means of six, five and four letter strings. Just type in the file name and the recorder and software do the rest. Can be used in the BiSync (IBM), BiPhase (Phase Encoded) or NRZ modes with suitable recorders and interfaces.

$190, wired and tested.
$160, Kit form.

Audio Cassette Interface (ACI) This is the phase encoding board used in the 3M3. Additional components on the board enable you to use audio recorders in the KC standard or the new PE 2400 (2400 baud) systems. Can also be used for Tarbell if you have an 8251 Intel I/O chip. Required if you use an audio cassette with the 2SIO(R) above.

$55, wired and tested.
$35, Kit form.

For 6800 Users: Software programs and I/O board for SWTP are under development. Limited software available now. Ask for 6800 data with $3.00 Documentation package. These programs will provide full software control.

CARTRIDGE AVAILABILITY: Cartridges are made by 3M, ITC, Wabash and others. They are available at all computer supply houses and most major computer service centers. We can supply them at normal current list prices.

"COMPUTER AID" and "UNIBOARD" are trademarks of the NATIONAL MULTIPLEX CORPORATION. The 3M Data Cartridges are covered by 3M Patents and Marks. "UNIBOARD" Patents Pending.

OVERSEAS: EXPORT VERSION — 220 V [illegible] Hz. Write Factory or: Megatron, 8011 Putzbrunn, Munchen, Germany; Nippon Automation 5-16-7 Shiba, Minato-Ku, Tokyo; Hobby Data, FACK 20012 Malmo, Sweden; G. Ashbee, 172 Ifield Road, London SW 10-9 AG.

For U.P.S. delivery, add $3.00. Overseas and air shipments charges collect. N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and BankAmericard accepted.

NATIONAL MULTIPLEX CORPORATION

3474 Rand Avenue, South Plainfield NJ 07080, Box 288, Phone (201) 561-3600 TWX 710-997-9530

A MICROPROCESSOR COURSE
FOR ENGINEERS AND PROGRAMMERS

FROM OSBORNE & ASSOCIATES, INC.
AUTHORS OF THE "INTRODUCTION TO MICROCOMPUTERS" SERIES OF BOOKS

This course teaches students how to design products around microprocessors.

This is the principal difference between a microcomputer and a minicomputer: a minicomputer is used as a vehicle for the execution of various programs; a microcomputer is used as a low cost substitute for combinatorial logic.

A microcomputer can be used, like a minicomputer, simply to execute programs; that is how microcomputers are used by computer hobbyists. Now the microcomputer user is only interested in the programming languages, and the microcomputer is packaged as a minicomputer-like system.

In sharp contrast, a logic designer who is using a microcomputer to replace combinatorial logic must look upon programming and digital logic as two interdependent aspects of microprocessor utilization.

Most microprocessor courses being taught at the present time do not accurately stress the interdependence of hardware and software. The student can go through an assembly language programming course, and the fact that a microcomputer assembly language has been taught leaves the student no closer to understanding how the microprocessor should be used. Unless assembly language is taught as an alternative to implementing transfer functions which could otherwise be implemented using combinatorial logic, students will not understand how to use microprocessors in a digital logic environment. The purpose of the course being taught by Osborne & Associates is to provide this direct comparison between hardware and programming. Students attending our course are taught how to decide whether hardware or instructions within a microcomputer system should be used in order to implement each step of any product. Emphasis is placed on giving the student a flexible understanding of whether to keep logic inside or outside the microcomputer system, how to configure the microcomputer system, and how to write programs that drive the configured microcomputer system.

Heavy emphasis is placed on classroom assignments; students are required to implement the same function using digital packages, instruction sequences within a microcomputer system, and combinations of the two.

TIME	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5
8:30 - 10:00 a.m.	SESSION 1 Single Signal Logic	SESSION 4 The Central Processing Unit	SESSION 7 Assembly Language Programming	SESSION 9 Subroutines and Interrupts	SESSION 11 Parallel Interfaces
			BREAK		
10:30 a.m. - 12:00	SESSION 2 Parallel Data Concepts	SESSION 5 Memory Organization	SESSION 8 Memory Addressing Modes	SESSION 10 The 8080A CPU	SESSION 12 Serial I/O
			LUNCH		
1:00 - 2:30 p.m.	SESSION 3 The Structure Of A Microcomputer	SESSION 6 Elementary Assembly Language Programming	HANDOUT 4	HANDOUT 4 (Cont'd) HANDOUT 5	HANDOUT 5 (Cont'd)
			BREAK		
3:00 - 4:30 p.m.	HANDOUT 1/HANDOUT 2	HANDOUT 3	HANDOUT 4 (Cont'd)	HANDOUT 5 (Cont'd)	HANDOUT 5 (Cont'd)
			EVENING		
	HANDOUT 1/HANDOUT 2 (Cont'd)	HANDOUT 3 (Cont'd)	HANDOUT 4 (Cont'd)	HANDOUT 5 (Cont'd)	HANDOUT 5 (Cont'd) HANDOUT 6

TIME	DAY 6	DAY 7	DAY 8	DAY 9	DAY 10
8:30 - 10:00 a.m.	SESSION 13 Memory Systems Configurations	SESSION 15 Programming Aspects Of Interrupts	SESSION 17 The Theory Of Direct Memory Access	COMPLETE HANDOUTS	An Overview of Microcomputer Systems
			BREAK		
10:30 - 12:00	SESSION 14 One-Shots & Interval Timing; The 8253 Programmable Timer	SESSION 16 Interrupt Handling Devices	SESSION 18 The 8257 Direct Memory Access Controller	COMPLETE HANDOUTS	An Overview Of Microcomputer Systems (Cont'd)
			LUNCH		
1:00 - 2:30 p.m.	HANDOUT 6 (Cont'd) HANDOUTS 7, 8 & 9	SESSION 16 (Cont'd)	COMPLETE HANDOUTS	COMPLETE HANDOUTS	An Overview Of Microcomputer Systems (Cont'd)
			BREAK		
3:00 - 4:30 p.m.	COMPLETE HANDOUTS	COMPLETE HANDOUTS	COMPLETE HANDOUTS	COMPLETE HANDOUTS	An Overview Of Microcomputer Systems (Cont'd)
			EVENING		
	COMPLETE HANDOUTS	COMPLETE HANDOUTS	COMPLETE HANDOUTS	COMPLETE HANDOUTS	

Course fee: $600 per student at our facilities
(10 students per class)
$6,000 per class, plus expenses, at your facility.

OSBORNE & ASSOCIATES, INC.
P.O. Box 2036
Berkeley, California 94702 • (415) 548-2805

CIRCLE INQUIRY NO. 18

LEGION: An Experiment in Artificial Intelligence

By Marlin Ellers

Most articles about artificial intelligence start out with an introduction to the subject, a definition of artificial intelligence, and an apology to the many people in the field who disagree with the definition which the author has just put forth. One of the main reasons for this is that although most people agree upon the meaning of the word "artificial" very few agree upon the meaning of "intelligence", particularly when the critter that is claiming to have it is a machine. In an attempt to avoid quarrelsome definitions and hair-splitting arguments, and also in order to keep this article from becoming tediously long, your author will simply ignore the question of what artificial intelligence is for the time being and describe LEGION and allow the reader to come to his own conclusions and definitions.

THE PROBLEM

We would like to build a program that is capable of recognizing characters which are drawn as a dot matrix (see Fig. 1). Furthermore we would like the program to be able to "learn" what a character is by showing it several examples of a character and telling it what kind of character it is.

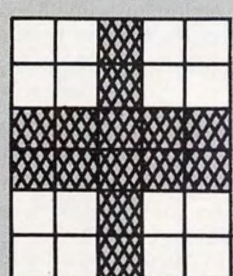


Figure 1. a 5x6 Dot Matrix Character (a plus sign)

LEGION

LEGION is the name of the program that the author wrote in a mixture of FORTRAN and Assembly Language in a couple of days that was an attempt to solve the above problem. It was not altogether successful, but it did have many interesting features and was quite fun to play with. Since LEGION was written quickly and sort of kluged together a general description is given rather than a listing.

The basic design decision was that LEGION would be made up of a number of small parts, each of which was capable of deciding what it thought a character was, and each of which could be punished or rewarded for coming to a right or wrong decision.

The idea was that the parts that arrived at correct decisions most often would have a stronger say in the total decision that LEGION would finally make when asked about a character and thus with a little practice LEGION could learn what a character was. At this point it was decided that the parts would be called "citizen" and when LEGION was shown a character each "citizen" would "VOTE" on what kind of character he thought it was, and LEGION would simply tally the votes and see who the winner was. In addition each citizen would have a "bank account" which was essentially a tally of how often the "citizen" was right. Each time that a "citizen" guessed correctly a dollar was credited to his account. If, on the other hand he was wrong in his guess, he would lose a dollar from his account. When election time came to the community LEGION followed the very democratic principle of "one dollar," "one vote" thus weighting the "good" voters more heavily than the "bad" ones.

When LEGION was finally implemented several previously unspecified parameters of the problem were chosen in such a way as to simplify the programming on a 16-bit word machine, and to simplify the problem of inputing characters. (Recall that being a learning machine requires that a given character may have to be shown to the program several times before it will learn it. This does not present any problem to a system that has a card reader, but when all that you have

is a teletype and each character must be typed and retyped into the machine there is a definite need to keep the total input needed to a minimum).

For example, it was decided that a character would consist of a 5 by 6 diode matrix so that one character would fit conveniently into two 16-bit words (with two bits to spare). Also it was decided that LEGION would only be required to decide between four characters; a plus, a minus, a slash, and a zero.

A "citizen" voted on a particular character in this manner: a subroutine "VOTE" was called which treated a citizen (which consisted of a 16-bit number called his social security number of I.D. number) as a set of 16 switches that told whether to perform or not to perform certain fixed manipulations of the two words of character. For example, the first bit of the I.D. number tells whether or not to rotate the two words of character by one bit, the second tells whether or not to "Exclusive Or" the I.D. number in with the character etc. These manipulations were chosen at random in an attempt to insure that different "citizens" would scramble up the character quite differently. When a "citizen" was done scrambling the character it simply threw out all but the last two bits which it took as a number from one to four and that was that "citizen's" guess for that character. **Note** that this method means that an individual "citizen" is totally incapable of learning anything, for given the same character over and over again he will always scramble it the same way and will thus always make the same guess as to what it is.

LEGION is an interesting example of a whole that is equal to more than the sum of its parts, where a society of inflexible unlearning individuals can work together to form a flexible learning body, just like a group of politicians (or worse yet, just the opposite).

It was decided the LEGION would start out as a colony of 1000 people each with an initial bank account of $15.00. Also since any "citizen" had a one out of four chance of guessing right on any one character it was decided that everyone would lose $1.00 for a bad guess and gain $3.00 for a correct guess thus keeping the total worth of the society at a steady level. Finally, in order to introduce new blood into society, it was decided that anyone that lost all of his money was clearly not worthwhile and would be replaced with a new individual who would start out with a fresh "bank account" of $15.00. (The inflation introduced by this arbitrary introduction of new money was simply ignored. After all, normal people are forced to live with it so why should LEGION have it any better?) It took about three days to write and debug LEGION and he was ready for the big test, could he learn ? ? ?

THE TEST

LEGION was first shown the 20 characters shown in Fig. 2 and asked to sort them into the four previously described classes. He did not do an overly impressive job, getting 90% wrong the first time through. The teacher (the author) decided to see if LEGION could learn a single character. After showing a Plus about 10 times LEGION finally decided that the picture was a Plus instead of a zero. (Yea! a Victory). He was then shown another Plus. After about 4 showings LEGION decided that the second character was a Plus

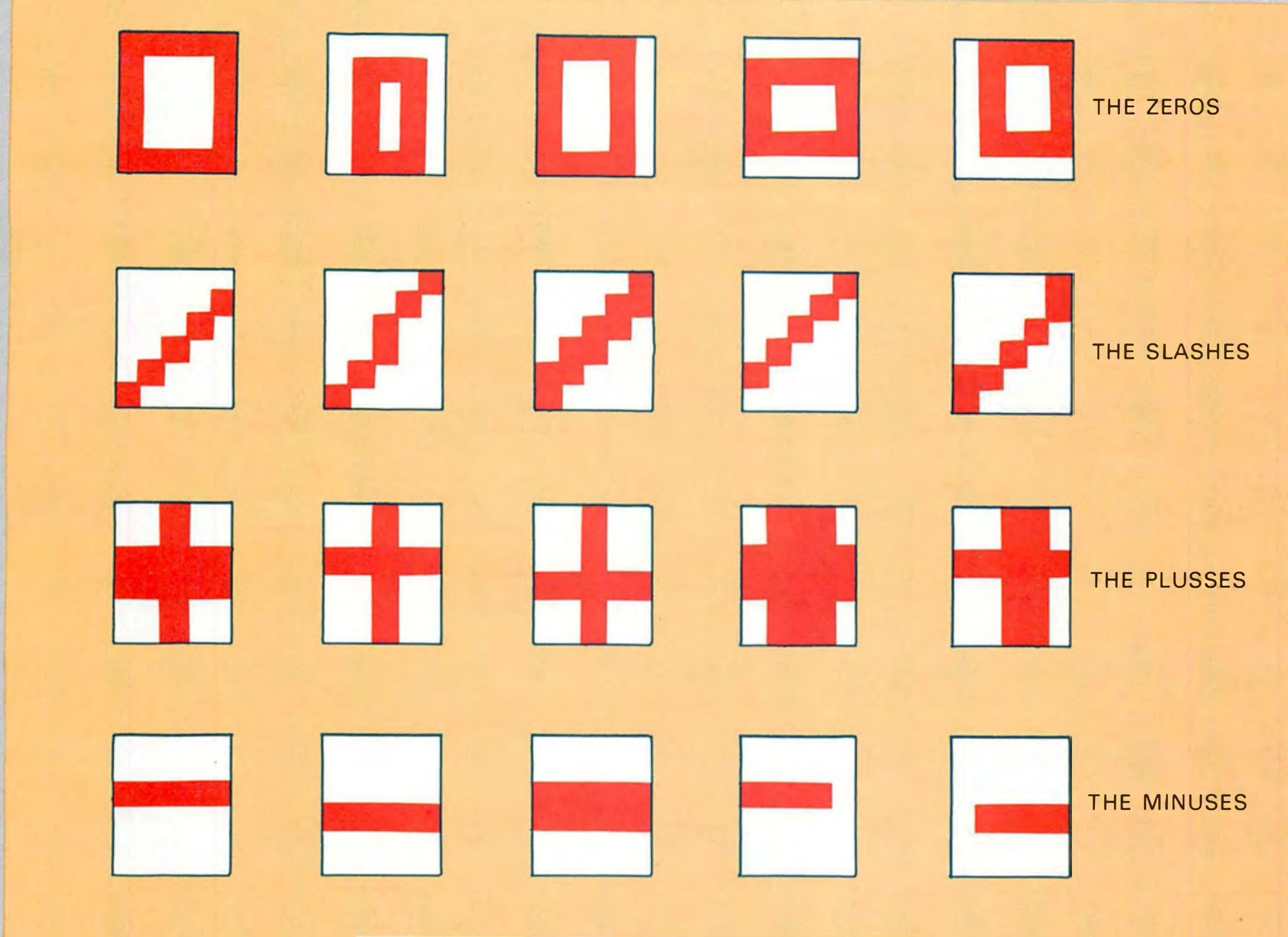
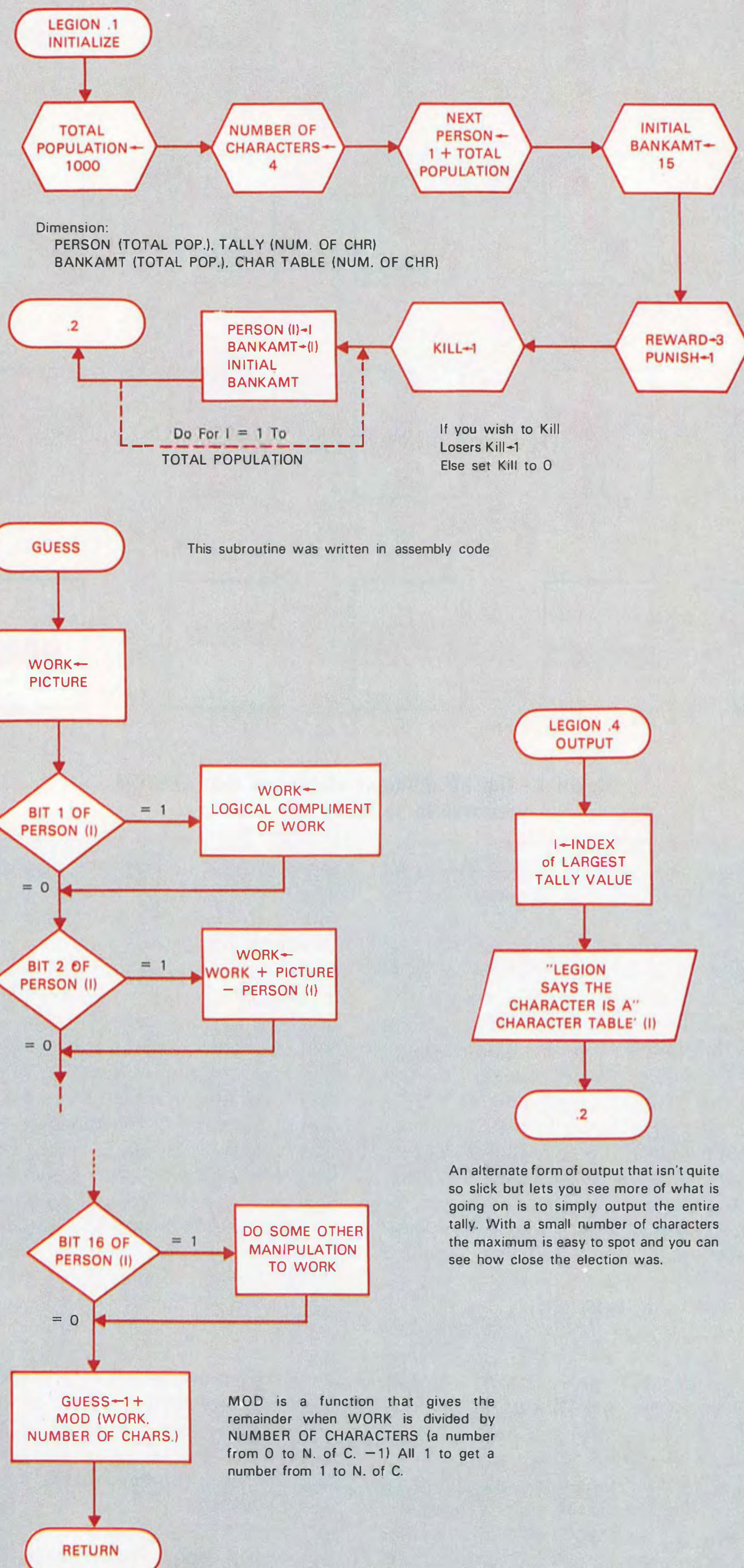


Figure 2. The 20 different characters that LEGION was trained to recognize.

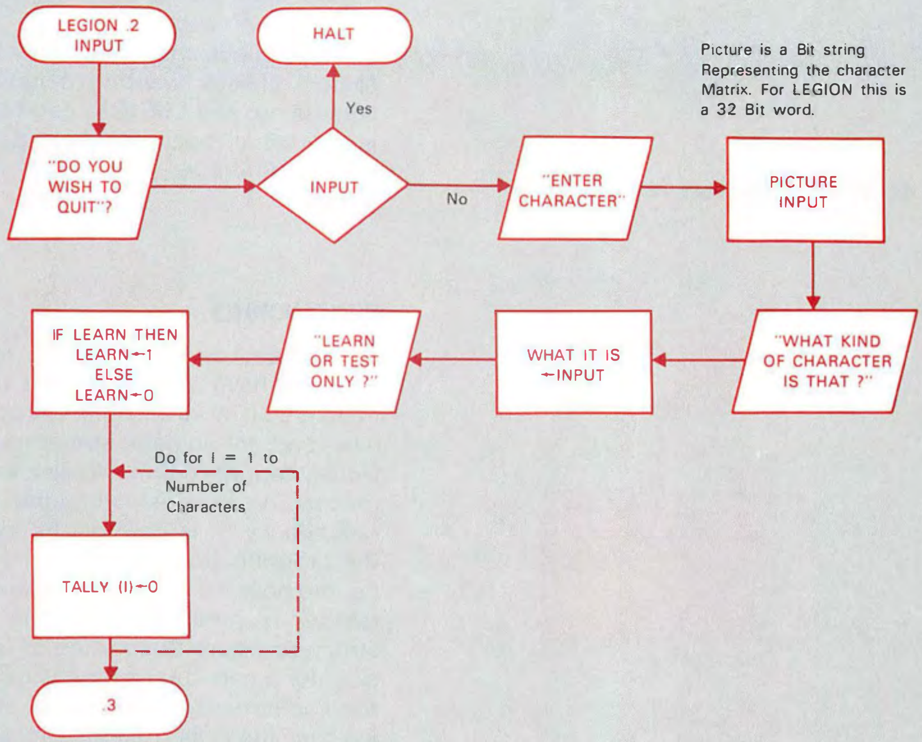
instead of a zero. (Yea! Another Victory). It was then reshown the first character, he guessed a zero (Tsk tsk you should remember longer than that). With one reshowing however LEGION remembered that the first one was a Plus (Yea! again). After a little drill LEGION learned the Minus, and in the process forgot the Pluses. By alternating between the three of them LEGION eventually learned them all. After several hours of similar trials, familiar to every teacher LEGION had made considerable progress from an initial 90% wrong to 75% right. However in spite of this progress the teacher was frustrated that it was taking so long to learn what seemed to be such a simple task. The decision was that the teacher was simply being too lenient, "spare the rod and all that." Too many individuals that were not really any good were staying alive. After all, one could stay alive with guessing no better than random. Well that was quickly cured. With a fairly simple program change the penalty for failure to guess correctly was upped to $2.00. The change that this brought over LEGION was simply amazing. Suddenly LEGION began to pick things up a little quicker and in short order was up to a perfect score, all 20 characters correctly identified.

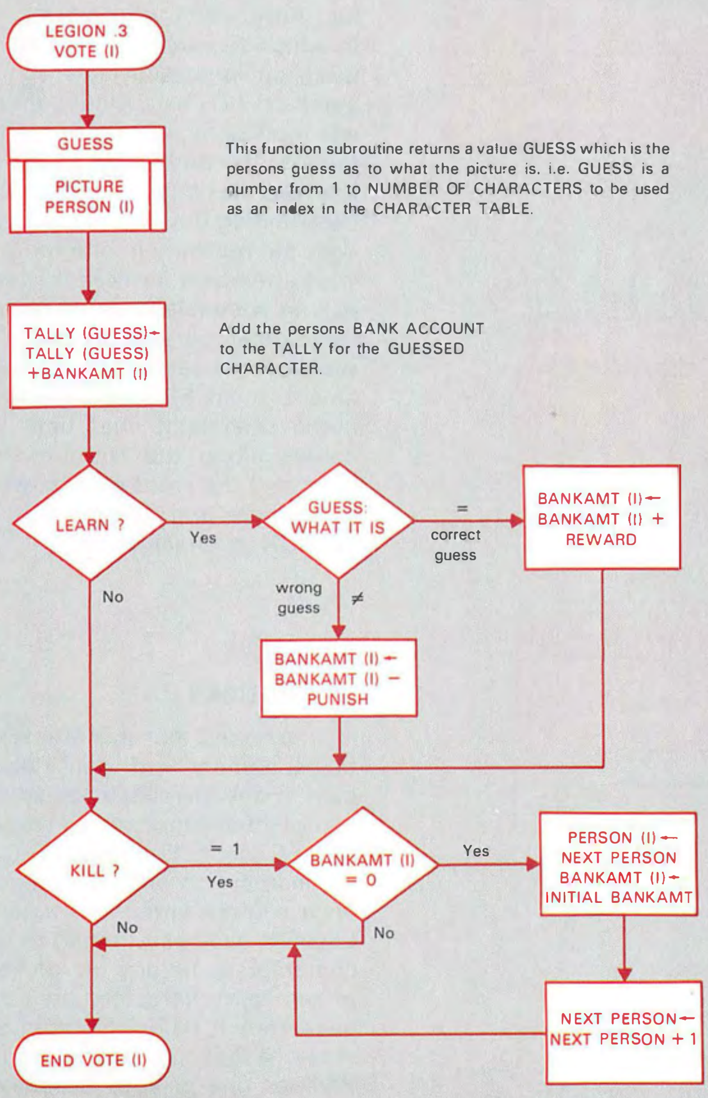
At this point LEGION's brains were dismantled via a core dump in order to do some statistical checking and some curiosity satisfying; such as who had the biggest bank account? Was there a "perfect citizen," i.e. one who just happened to always be right? How many citizens were mediocre as compared to the rich ones, i.e. were the rich an elite few who controlled most of society, or did the masses do most of the decision making? The surprising thing was that due to a programming bug there were citizens who had negative bank accounts! This was because under the old punishment system of up $3.00 down $1.00 it was decided to throw out someone when his bank account hit zero, so a test for zero was made. However under the new up $3.00 down $2.00 it was possible to have one's bank account go from +$1.00 to −$1.00 in one jump and thus get around the test for zero. This may have been partly responsible for the increased learning ability after the change, after all a citizen that consistently guesses wrong is just as valuable as one that is consistently right. The original classification as to "good" and "bad" citizens was wrong and unfair. A derelict who squanders his money performs valuable service to his society, and is cherished by the Creator (the author) as much as the model citizen who frugally tries to vote correctly in every election! Quite a revelation that may have far reaching consequences when applied to a society of actual persons. The old order was overthrown, a new paradigm came into being, a new method of introducing new blood had to be introduced. Was random death the answer? Is that the reason that all people must die? As he stood at that moment LEGION handled everything that had been thrown at him. Could he handle more? There was much interest in progressing further to see just what LEGION's limits were however at this point the powers

Figure 3. Detail Flowchart for LEGION



Continued

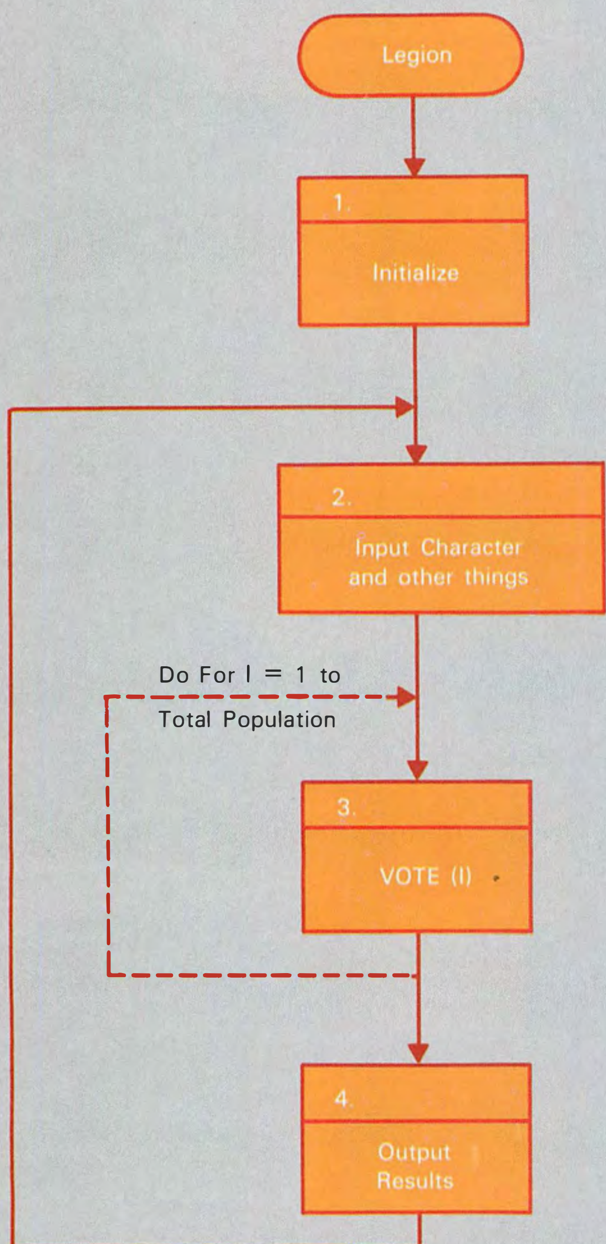

LEGION .2
INPUT
HALT
Yes
"DO YOU
WISH TO
QUIT"?
INPUT
No
"ENTER
CHARACTER"
PICTURE
INPUT
Picture is a Bit string
Representing the character
Matrix. For LEGION this is
a 32 Bit word.
IF LEARN THEN
LEARN←1
ELSE
LEARN←0
"LEARN
OR TEST
ONLY ?"
WHAT IT IS
←INPUT
"WHAT KIND
OF CHARACTER
IS THAT ?"
Do for I = 1 to
Number of
Characters
TALLY (I)←0
.3


LEGION .3
VOTE (I)
GUESS
PICTURE
PERSON (I)
This function subroutine returns a value GUESS which is the
persons guess as to what the picture is. i.e. GUESS is a
number from 1 to NUMBER OF CHARACTERS to be used
as an index in the CHARACTER TABLE.
TALLY (GUESS)←
TALLY (GUESS)
+BANKAMT (I)
Add the persons BANK ACCOUNT
to the TALLY for the GUESSED
CHARACTER.
LEARN ?
Yes
No
GUESS:
WHAT IT IS
=
correct
guess
BANKAMT (I)←
BANKAMT (I) +
REWARD
wrong
guess
≠
BANKAMT (I)←
BANKAMT (I) −
PUNISH
KILL ?
= 1
Yes
No
BANKAMT (I)
= 0
Yes
No
PERSON (I)←
NEXT PERSON
BANKAMT (I)←
INITIAL BANKAMT
NEXT PERSON←
NEXT PERSON + 1
END VOTE (I)

that be stepped in and asked what was going on and LEGION just barely jumped out the window in time to avoid being caught.

It has since that time, been a busy year, with customers always wanting a new operating system or something, and LEGION, good friend and kluge that he is, has never been resurrected, never been pushed to his full potential.

Figure 4. Overall Flowchart for LEGION



CONCLUSIONS

The question still remains, "Is LEGION intelligent?" He does have several on the tell-tale signs. For one thing LEGION is flexible enough to learn something new and to unlearn something that has been previously taught to him. He also exhibits a feature that is uncommon in most computer programs and that is redundancy. It is possible to erase great quantities of the program (the citizenry) and although he will lose his memory he is able to relearn everything. This last feature is reminiscent of the experiments in brain structure where the portion of the brain that is responsible for a particular action is isolated and removed but the impairment of function is only temporary, indicating that the brain has built in a certain redundancy of function. But is LEGION really intelligent? He can't tie his shoes, can't write poetry, can't even weave a web in which to catch flies for dinner and would surely receive an impossibly low score in the Stanford-Binet standard I.Q. test. Under those standards LEGION is not intelligent; but then neither are some of my best friends. The problem is in the definition of intelligence, it always has been. One test that has been put forth for determining if a machine possesses intelligence is that with the machine in one room and a person in another with a teletype for communication between them, the person is unable to tell if he is talking to a machine or to another person. LEGION would fail this test as would most readers if the person testing were only able to speak Mandarin Chinese. For these reasons the author maintains that until an acceptable definition comes along the Humpty-Dumpty criterion will be used and the word means what I want it to mean, nothing more and nothing less and under my definition LEGION is intelligent.

SUGGESTIONS

It is hoped that this article will inspire some readers to experiment with such learning programs on their own. If anyone should do so they may wish to try this test of intelligence which was going to be the next one for LEGION before he was so rudely interrupted. Does the learning on one character set enhance the learning on a slightly larger character set? In particular after LEGION has been trained to recognize the first twenty characters is he any more likely to be correct on a set of ten test characters than he would be if they had been shown before learning anything? Does he apply what he has previously learned to a new problem? Perhaps one of you can answer that.

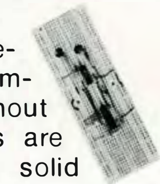
Solder is for sissies.

Are. you still designing circuits the old fashioned way? Soldering and unsoldering? Burning up valuable IC's? If so, it's time you made a switch to solderless breadboarding. E&L Instruments offers a complete line of breadboarding products from sockets to the most complete systems for professional use. Here are five ways you can go solderless . . .

Breadbox III

An economical approach to a large breadboarding area. Three SK-10 sockets, four 5-way binding posts, two BNC connectors, six BP-22 solderless breadboarding pins. $72.25

SK-10 Socket

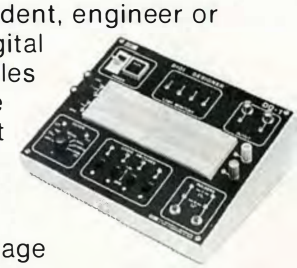


This precision solderless breadboarding socket will pay for itself in one afternoon. Integrated circuits and discrete components insert directly, without adapters. All interconnections are made with 20 to 26 gage solid conductor wire. $17.75.

Op-Amp Designer #OA-2K

Powerful design tool for linear work and CMOS digital circuitry. SK-10 socket. Heavy current capacities on all three power supplies [+/– 15V & +5V DC]. Sensitive null detector and a full function generator with external frequency adjustment. All interconnections are solderless. $120.00 in kit form.

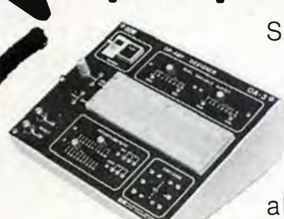
Digi-Designer #DD-1K



A must for every student, engineer or technician doing digital circuit design. Handles both IC and discrete components without soldering. Connections to the SK-10 socket are made using any solid 22 gage insulated wire. The DD-1K includes a clock, power supply, logic lights, pulsers, logic level switches and much more. All circuitry is now in one circuit board. $77.75 in kit form.

All of these products are now available at your local computer store. Prices shown are suggested resale U.S.A.

Op-Amp Designer #OA-3K



SK-10 socket. Plus and minus variable supplies (1 to 15V). This adjustable feature makes the unit exceptionally flexible, allowing it to be more useful for odd voltage requirements. The fixed 5 volt supply allows interfacing with TTL circuitry. $102.50 in kit form.

E&L INSTRUMENTS, INC.
61 First Street, Derby, Conn. 06418
(203) 735-8774 Telex No. 96 3536

Dealer inquiries invited.

CIRCLE INQUIRY NO. 8

MICROCOMPUTER STOCK OPTIONS

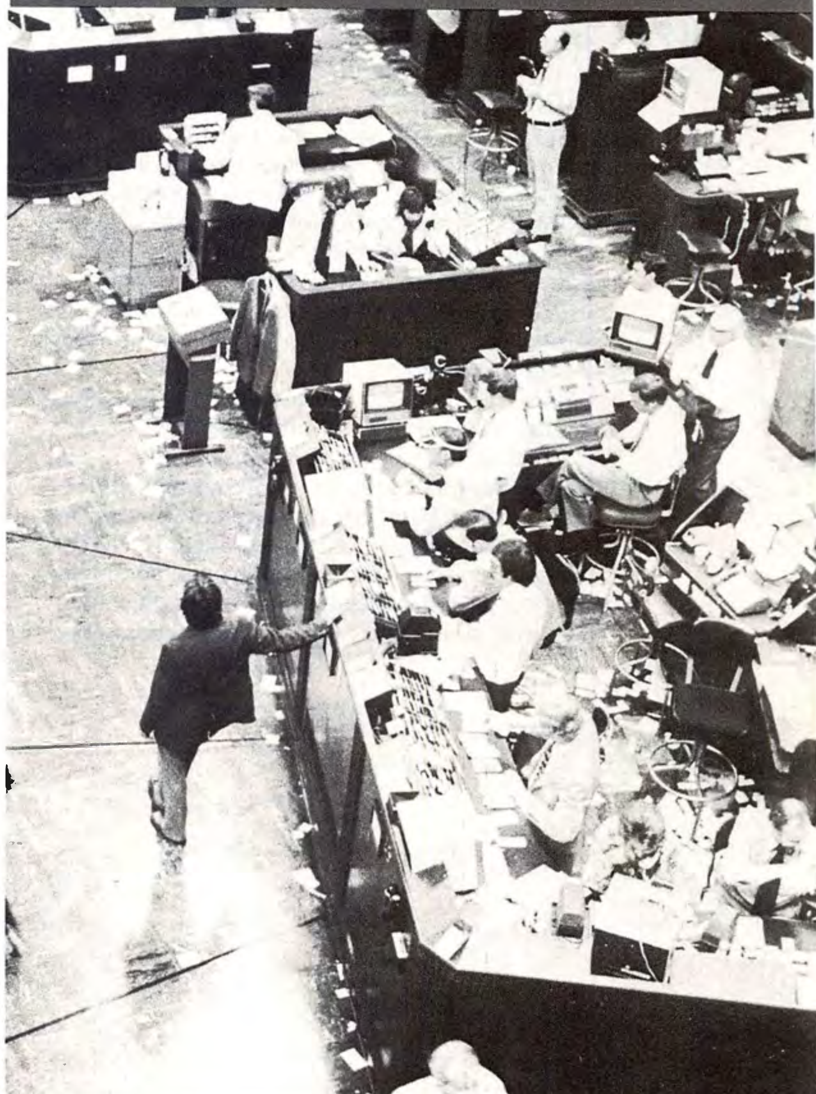
TO *HEDGE* OR NOT TO *HEDGE* — THE OPTION IS YOURS

FOREWORD

This stock option program is written in Processor Tech's 5K BASIC for the 8080 Microcomputer and requires 4.2K bytes of memory storage space.

INTRODUCTION

If you have an interest in the stock market or in listed call options, here's a way to use that interest to help pay for your home-brew system, buy some extra memory, or answer the inevitable question — "What ***are*** you going to do with your own computer?" The program described in this article calculates the net profit and rate of return for a simultaneous purchase of stock and sale of a call option, known generally as a ***hedge*** and specifically as covered writing.



Until recently a buyer of *calls* had to go to a *call* broker. This method was relatively slow and resale was difficult.

By Edward Christianson

WHAT IS AN OPTION?

A call option, or ***call***, is the right to buy a number of shares of an underlying stock at a fixed price, called the strike or exercise price, until a fixed date, called the expiration date. Until recently, a buyer of ***calls*** had to go to a ***call*** broker who would find someone willing to offer (or write) a ***call*** at or near the price desired by the buyer. This method of buying and writing ***calls*** was relatively slow, and once a ***call*** was purchased, resale was difficult because there was no ready market for them.

In April, 1973, the Chicago Board Options Exchange (CBOE) began auction trading of ***calls***, and a whole new world of investment strategies began with it. The CBOE's approach was simple. Each ***call*** would be in units of 100 shares, the strike price would be in increments of 5, 10, or 20 dollars per share (depending on the price of the underlying stock) and the expiration date would be fixed at one of four months of the year — January, April, July, and October. However, only three expiration periods would be traded at any one time. In addition, a liquid aftermarket would be maintained through a system of competing market makers. This meant that a ***call*** could be bought and sold minutes, hours, days, or weeks later depending on the buyer's assessment of market conditions. This idea was so successful that trading in ***call*** options grew from ten underlying stocks on the CBOE to nearly 200 underlying stocks on the CBOE, American Stock Exchange (AMEX), the Philadelphia, Baltimore and Washington (PHLX), the Pacific Coast Exchange (PCE), and Midwest Stock Exchange (MSE).

HOW OPTIONS WORK

To understand why CBOE type ***calls*** are so successful, the motives of buyer and writer alike need to be examined. First, we need a brief explanation of the interaction of stock and option prices. Using the stock prices of a hypothetical company, the cost of a ***call*** with a $50 strike price at various stock prices with 6 months to go, 3 months to go, and at expiration is shown in Table 1.

Table 1 — Cost of $50 Strike *Call* Option

Stock Price	6 Months to go	3 Months to go	At Expiration
40	$ 50	$ 37	$ 0
45	200	150	0
50	500	350	0
55	800	650	500
60	1,100	1,050	1,000
65	1,500	1,500	1,500

For example, if the stock was at $45 per share, a 6 month ***call*** would cost $200. If the stock was at $55 per share, that same ***call*** would cost $800.

The cost of a ***call*** can be divided into two parts:

1. ***Premium*** which represents the value for the time left until expiration. It decreases slowly in the first 5 to 6 months, then decreases much more rapidly during the last 3 months of life. Fox example, using the $50 stock price in Table 1, a ***call*** may be worth $500 with 6 months to go, $350 with 3 months, and be worthless at expiration. Figure 1 shows a relationship between ***premium*** and time.

   There are other factors which influence ***premiums***. The options of more volatile stocks, which move up quickly during rallies, command higher ***premiums.*** The supply and demand for options on the floor of the exchange also influences the ***premium.*** Also, a significant dividend from an "income" stock represents added income to the writer. ***Call*** buyers recognize this income, and adjust their bids on ***calls*** accordingly.

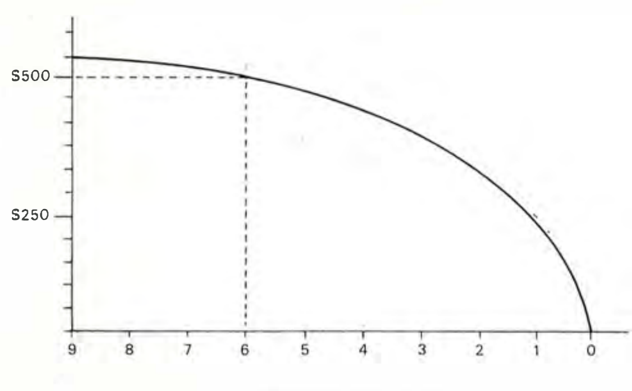


Figure 1. *Premium* vs Time

2. ***Intrinsic value*** is the stock price minus the strike price and is always positive. The relationship of intrinsic value in the cost of a call can best be explained in Figure 2. For example, notice that when the stock price is $55, the option cost is $800. The ***intrinsic value*** is $500 and the ***premium*** is $300.

In a sharply rising market, buying *calls* is an excellent strategy.

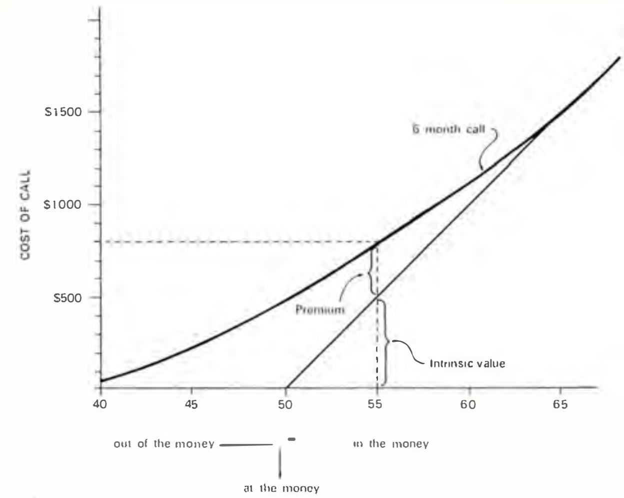


Figure 2. $50 Strike *Call* — Prices vs Stock Price

Several items should be noted here. First, as the stock price increases, the ***intrinsic value*** increases, dollar for dollar, once the stock price has risen above the strike price. Secondly, the ***premium*** virtually disappears when the stock price is 25% to 30% above its strike price.

When the stock price is above the strike price, the ***call*** is said to be "in the money." When the stock price is below the strike, the ***call*** is "out of the money." When the stock price is equal or nearly equal to the strike, the ***call*** is "at the money."

Using the prices from Table 1, we will calculate the profits, the capital requirements, and the return on investment for a buyer, a naked writer, and a covered writer. A naked writer is one who writes (offers) the option, but does not own the underlying stock. A covered writer owns the stock when the call is originally written.

THE OPTION BUYER

In this example, assume the stock is at $50 per share, and a 6 month $50 ***call*** is purchased for $500. Brokerage fees, dividends, and margin interest costs are ignored.

Stock Price	Value of Call	Profit (Loss)	Capital Required	Return %
40	$ 0	$-500	$500	-100
45	0	-500	500	-100
50	0	-500	500	-100
55	500	0	500	0
60	1,000	500	500	100
65	1,500	1,000	500	200

Table 2 — Profit or Loss at Expiration — Buyer

The buyer risks 100% of his capital. In fact, the stock must rise 10% (from 50 to 55) before he breaks even. After that, however, the rewards are quite dramatic. A 20% rise (from 50 to 60) will not only cover his costs, but will result in a 100% profit. In a sharply rising market, buying ***calls*** is an excellent strategy.

. . . Not for the beginner in options.

THE NAKED WRITER

In this example, the stock is at $50 per share, and a $50 strike ***call*** is written with $500 in proceeds to the writer. Again, brokerage, dividends, and margin interest are ignored.

A naked writer is required to put up at least 30% of the cost of delivering the shares. In addition, when the stock price is above the strike price, he must put up additional capital, dollar for dollar for each share for which he is naked. For example, when the stock is $5 above the strike, he would be required to put up another $500 in capital. On the other hand, if the stock was below the strike price, his capital requirements would be lowered, again dollar for dollar for each share. If the stock was $5 below the strike, the capital required would be lowered by $500. This process is called ***marking*** to the market.

Stock Price	Value of Call	Profit (Loss)	Capital Required	Return %
40	$ 0	$ 500	$ 500	100
45	0	500	1,000	50
50	0	500	1,500	33
55	500	0	2,000	0
60	1,000	-500	2,500	-20
65	1,500	-1,000	3,000	-33

Table 3 — Profit or Loss at Expiration — Naked Writer

The naked writer has an excellent return in a declining market. On the other hand, the capital requirements increase very rapidly because of the ***marking*** to the market requirements. A sudden, sharp rally could cause the naked writer to run out of capital. At this point, the brokerage firm would close out his position, resulting in a severe loss to the writer.

THE COVERED WRITER

In this example, the writer sets up his position by simultaneously buying the stock (100 shares at $50 per share) and offering the 6 month option, receiving $500 as proceeds. In any ***hedge*** transaction, the capital required is offset by the proceeds of the sale. Thus, the net capital required is $4,500 (5000—500). At or near expiration, two alternatives are possible.

Alternative 1. Allow the ***call*** to be exercised and deliver the stock. If the ***call*** is not exercised, the stock may be sold or another ***call*** may be offered against it. For purposes of this example, assume the stock is delivered or sold.

	OPTION		STOCK			
Stock Price	Value of Call	Profit (Loss)	Value	Profit (Loss)	Capital Required	Return %
40	$ 0	$500	$4,000	$-1,000	$4,500	-11
45	0	500	4,500	-500	4,500	0
50	0	500	5,000	0	4,500	11
55	a	500	b	0	4,500	11
60	a	500	b	0	4,500	11
65	a	500	b	0	4,500	11

Table 4 — Profit or Loss at Expiration — Covered Writer

Result a. To the writer, the value of the ***call*** if it is exercised is irrelevant — he has made his $500 profit.

Result b. Since he delivers the stock at $50 per share, his proceeds will be $5,000, regardless of the current market price of the stock.

Alternative 2. Close out the position before expiration by buying back the option and selling the stock.

	OPTION		STOCK				
Stock Price	Value of Call	Profit (Loss)	Value	Profit (Loss)	Profit (Loss)	Capital Required	Return %
40	$ 0	$ 500	$4,000	$-1,000	$-500	$4,500	-11
45	0	500	4,500	500	0	4,500	0
50	0	500	5,000	0	500	4,500	11
55	500	0	5,500	500	500	4,500	11
60	1,000	-500	6,000	1,000	500	4,500	11
65	1,500	-1,000	6,500	1,500	500	4,500	11

Table 5 — Profit or Loss from Early Closeout — Covered Writer

The net results are the same but there are reasons for choosing the second alternative over the first as we shall see later.

The covered writer limits his profit somewhat, but in return, he has more protection against loss. Covered writing strategies work best in flat or rising markets.

LEVERAGE

Leverage is using a small amount of capital to control a larger amount of capital. For example, a $500 ***call*** controls $5,000 worth of stock, a ***leverage*** ratio of 10 to 1. Though covered writing is the most conservative of the three methods, through increased ***leverage***, the return can be substantially increased. The covered writer uses a margin account which allows him to borrow capital (current 50%) to set up a position. In

return, the brokerage firm charges interest, called ***margin*** interest, on the capital borrowed, usually at not less than ½% above the prime rate.

THE COVERED WRITER—MARGINED

Next, we shall re-examine the same covered writing ***hedge*** using ***margin***, again excluding brokerage fees, dividends, and ***margin*** interest. The capital required is 50% of the stock purchase minus the proceeds from the sale of the option. Assume the position is closed out as in Alternative 2.

Stock Price	Value of Call	Profit (Loss)	Value	Profit (Loss)	Profit (Loss)	Capital Required	Return %
40	$ 0	$ 500	$4,000	$-1,000	$ -500	$2,000	-25
45	0	500	4,500	-500	0	2,000	0
50	0	500	5,000	0	500	2,000	25
55	500	0	5,500	500	500	2,000	25
60	1,000	-500	6,000	1,000	500	2,000	25
65	1,500	-1,000	6,500	1,500	500	2,000	25

Table 6 —Profit or Loss from Early Closeout — Margined Covered Writer

Note that the return is now 25% instead of 11%. However, the percentage loss possible is also increased.

Choose a broker who specializes in options, who understands *hedges* . . . and recommends suitable strategies in tune with your money.

BROKERAGE FEES

During the previous examples, we have ignored brokerage fees, dividends and ***margin*** interest. However, in the real world, they represent 30% or more of the total profit. Since May, 1975, brokers' fees have been negotiable, with the better customers getting larger discounts from the old rate structure. A recent development is the rise of the "discount" brokerage firm. Fully regulated, they do not offer market research nor recommendations, but merely execute your orders on the stock exchanges at discounts of 30% or more from the previous rate structure. In addition, many stocks, some of which have listed ***call*** options, are traded directly between brokerage firms as well as the floor of the major exchanges. Direct trading in this manner is known as over the counter (OTC) or the third market. Commissions are less on the OTC than on the major exchanges.

THE PROGRAM OUTPUT

The best way to demonstrate the effect of brokerage fees is to run our ***margined*** covered writer example through the program. The output is divided into four sections — Heading, Cost, Proceeds, and Return.

The Heading section gives the date of the evaluation, and restates the ***hedge*** — number of shares bought and number of options sold.

The Cost section includes the cost of the shares, the brokerage fees to buy the shares (labeled IN), to deliver the shares (labeled OUT), and to sell the option (labeled OPTION), and the ***margin*** interest cost. The Proceeds section includes the money received from delivering the shares at the strike price, the money from the sale of the options, and dividends (if any).

The Return section includes the net profit (total proceeds minus total cost), the net capital required, the percent net return, the number of days to expiration, the annual rate of return and the breakeven point. The breakeven point is the price of the stock (at expiration) where no profit will be made. It is computed by subtracting net profit from the original purchase cost.

Figure 3 is sample output of our margined covered writer using the New York Stock Exchange brokerage rates, and Figure 4 uses the OTC rates.

```
READY
RUN

INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)

?12,16,76,1,7.50
ØUTPUT TØ BE 32 ØR 64 CHARACTERS?
?64
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?1,.10
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?100,50,1
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?6,50,500

CØVERED WRITE NØ 1   12   16   76
BUY 100   SHARES AT 50
SELL 1     6   50 'S AT $ 500 EACH

        CØST                 PRØCEEDS            RETURN
YØUR $ 2500             STØCK   5000          $ 312 ØN
LØAN $ 2500             ØPTIØN   500           2188 IS
BRØKER'S                DIVIDEND 20            14.2 %
   IN       51                                 186 DAYS
  ØUT       51                                 = 27.8 %
 ØPTIØN     26                                 ANNUALLY
MARGIN      80                                 BRK-EVEN
        -----                  -----           IS
TØTALS 5208                     5520           $ 46.8
MØRE? (1=YES)
?1
```

Figure 3. Sample Using NYSE Brokerage Rate

In any *hedge* transaction, the capital required is offset by the proceeds of the sale.

```
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?2,.10
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?100,50,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?6,50,500

CØVERED WRITE NØ 2   12   16   76
BUY 100   SHARES AT 50
SELL 1     6   50 'S AT $ 500 EACH

        CØST                 PRØCEEDS            RETURN
YØUR $ 2500             STØCK   5000          $ 366 ØN
LØAN $ 2500             ØPTIØN   500           2134 IS
BRØKER'S                DIVIDEND 20            17.1 %
   IN       25                                 186 DAYS
  ØUT       25                                 = 33.5 %
 ØPTIØN     26                                 ANNUALLY
MARGIN      78                                 BRK-EVEN
        -----                  -----           IS
TØTALS 5154                     5520           $ 46.3
MØRE? (1=YES)
?0


STØP IN LINE 145

READY
```

Figure 4. Sample Using OTC Brokerage Rate

The annual rate of return will be increased if the months until the premium is gone can be shortened.

COMPOUNDING YOUR PROFITS

Another way to increase ***leverage*** is by compounding your profits. In the case of the covered writer, compounding is accomplished by taking the profit as soon as it becomes available and immediately reinvesting it, rather than waiting until expiration. The effect of compounding can best be demonstrated through a measure called the annual rate of return which is calculated as follows:

$$\text{ANNUAL RATE OF RETURN} = \text{rate of return \%} \times \frac{\text{12 months}}{\text{months until premium is gone}}$$

In our previous example, the

ANNUAL RATE OF RETURN = 25% x 12/6 = 50%

The annual rate of return will be increased if the months until the ***premium*** is gone can be shortened. Referring again to Table 1 or Figure 2, notice that the ***premium*** disappears when the stock price rises high enough. Suppose our ***margined*** covered writer notices 3 months later that the stock is now at $65, and the option cost is $1,500. He can close out his position now, and pocket the $500 profit. Look what happens to the annual rate of return.

ANNUAL RATE OF RETURN = 25% X 12/3 = 100%

He is now free to evaluate market conditions and enter into a new ***hedge.***

RUNNING THE PROGRAM

You will need the latest option quotations. Either the Wall Street Journal (daily) or Barron' Weekly is a complete, and reliable source. The program requires input in the following order:

Option		Sales (100s)	Open Int.	High	Low	Last	Net Chg.	Stock Close
N C R	Jan25	4	315	11	10⅝	11	+ 1½	36¼
N C R	Jan30	290	1398	6¼	4⅞	6¼	+ 1	36¼
N C R	Jan35	1073	5905	2	1⅜	1 13-16	+7-16	36¼
N C R	Apr30	51	434	7¼	6	7¼	+ 1	36¼
N C R	Apr35	751	2598	3⅜	2 9-16	3⅜	+7-16	36¼
N C R	Jul30	14	88	7⅜	6¾	7¼	+ ¼	36¼
N C R	Jul35	226	778	4¼	3½	4	+ ⅜	36¼
N Semi	Feb25	2867	3892	5⅛	4	4¼	......	27⅞
N Semi	Feb30	5872	11712	2⅛	1 9-16	1 11-16	−1-16	27⅞
N Semi	Feb35	3182	12388	¾	½	½	−1-16	27⅞
N Semi	Feb40	578	8502	3-16	⅛	⅛	−1-16	27⅞
N Semi	Feb45	228	4854	⅛	1-16	1-16	......	27⅞
N Semi	Feb50	3	5094	1-16	1-16	1-16	......	27⅞
N Semi	May25	788	2261	6¼	5⅜	5½	− ¼	27⅞
N Semi	May30	2287	6933	3⅝	2 15-16	3	− ⅛	27⅞
N Semi	May35	1800	6958	1¾	1⅜	1 7-16	− ⅛	27⅞
N Semi	May40	1457	5188	13-16	9-16	11-16	.....	27⅞
N Semi	Aug25	588	853	7¼	6¼	6⅜	− ⅛	27⅞
N Semi	Aug30	1233	1712	4½	3¾	3⅞	− ⅛	27⅞
Nw Air	Jan25	56	411	5⅝	5	5¼	− ⅛	29¾
Nw Air	Jan30	551	3027	1⅝	1 3-16	1 3-16	−3-16	29¾
Nw Air	Jan35	225	2280	5-16	⅛	⅛	− ⅛	29¾
Nw Air	Apr25	33	143	6	5⅝	5⅞	+ ⅛	29¾
Nw Air	Apr30	250	1662	2 11-16	2⅜	2⅜	− ⅛	29¾
Nw Air	Apr35	167	1377	15-16	¾	13-16	−1-16	29¾
Nw Air	Jul25	4	131	6¾	6¾	6¾	− ¼	29¾
Nw Air	Jul30	131	606	3½	3	3	− ⅛	29¾
Occi	Feb15	1359	2410	7¼	5¾	6⅞	+ ⅝	21⅞
Occi	Feb20	7458	21020	2 7-16	1 9-16	2 1-16	+ ¼	21⅞
Occi	May15	622	2146	7⅜	6⅛	7	+ ⅝	21⅞
Occi	May20	2807	10179	3¼	2 3-16	2⅞	+7-16	21⅞
Occi	Aug15	330	437	7½	6¼	7¼	+ ½	21⅞
Occi	Aug20	1329	2776	3¾	2¾	3⅜	+7-16	21⅞
Pennz	Jan25	57	416	7½	7	7¼	− ¼	32⅛
Pennz	Jan30	458	2815	2⅞	2⅛	2 7-16	−1-16	32⅛
Pennz	Jan35	290	3667	⅜	3-16	3-16	−3-16	32⅛
Pennz	Apr25	37	44	7⅝	7⅛	7½	− ⅛	32⅛
Pennz	Apr30	358	1615	3¾	3	3¼	......	32⅛
Pennz	Apr35	316	1939	1 3-16	⅞	1	− ⅛	32⅛
Pennz	Jul25	24	82	7¾	7	7¾	+ ¼	32⅛
Pennz	Jul30	56	533	4¼	3⅞	3⅞	− ⅛	32⅛
Pepsi	Jan70	162	191	13⅜	11⅛	11⅛	− ⅞	80¾
Pepsi	Jan80	246	548	4¼	2¼	2½	− ¾	80¾
Pepsi	Jan90	237	347	⅜	3-16	5-16	......	80¾
Pepsi	Apr70	63	86	14½	13	13⅞	+ 1¼	80¾
Pepsi	Apr80	111	267	6	4½	4⅝	− ¼	80¾
Pepsi	Apr90	34	400	2	1⅜	1⅜	......	80¾
Pepsi	Jul80	19	66	7¼	5¾	5¾	− ½	80¾
Pepsi	Jul90	16	110	2⅝	2⅜	2⅜	......	80¾
Polar	Jan30	2934	2907	10¾	7½	8¾	+ 1⅜	38⅞
Polar	Jan35	7460	8838	5⅞	3⅜	4¼	+ 1	38⅞
Polar	Jan40	16[illegible]81	23304	2 1-16	⅞	1 7-16	+9-16	38⅞
Polar	Apr35	2442	4858	7⅛	5	5⅝	+ ⅞	38⅞
Polar	Jul35	642	2421	8	6	6¾	+ ¾	38⅞
Polar	Apr40	6543	10529	5¼	2⅜	2⅞	+9-16	38⅞
Polar	Jan45	2836	14935	9-16	3-16	7-16	+5-16	38⅞
Polar	Apr45	2988	6349	1 15-16	15-16	1⅜	+7-16	38⅞
Polar	Jul40	1720	3445	5	3⅜	4	+ ¾	38⅞
R C A	Jan20	386	290	7⅜	5⅛	7⅛	+ 2	26⅝
R C A	Jan25	3547	7624	2⅜	1	2⅛	+ 1⅛	26⅝
R C A	Jan30	1375	12628	¼	1-16	3-16	+1-16	26⅝
R C A	Apr20	103	193	7⅜	5⅜	7¼	+ 1⅞	26⅝
R C A	Apr25	1803	5112	3⅛	1 13-16	2⅞	+1⅛	26⅝
R C A	Apr30	1714	7316	15-16	7-16	13-16	+5-16	26⅝
R C A	Jul20	108	257	7½	5¾	7¼	+ 1⅝	26⅝
R C A	Jul25	623	2322	3⅝	2 5-16	3½	+ 1⅛	26⅝
Raythn	Feb50	8	67	12	11⅝	11⅞	+ 2⅜	61⅛
Raythn	Feb60	116	802	4¼	3¼	3¼	+ ⅜	61⅛
Raythn	Feb70	6	672	5-16	¼	5-16	+1-16	61⅛
Raythn	May60	23	205	5¾	5	5⅛	+ 1	61⅛
Raythn	May70	81	336	1½	1 1-16	1¼	+ ⅛	61⅛
Raythn	Aug50	4	.....	12½	12½	12½	......	61⅛
Raythn	Aug60	5	47	6¾	6½	6¾	+ 1½	61⅛

Example. Excerpt from Barron's Option Quotation

```
RUN

INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)

?12,10,76,1,7.5
ØUTPUT TØ BE 32 ØR 64 CHARACTERS?
?64
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)
?101,.08
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?200,38.87,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?4,35,562

CØVERED WRITE NØ 101   12   10   76
BUY 200   SHARES AT 38.87
SELL 2     4   35 'S AT $ 562 EACH

        CØST                 PRØCEEDS              RETURN
YØUR $ 3887             STØCK  7000             $ 169 ØN
LØAN $ 3887             ØPTIØN  1124            2944 IS
BRØKER'S                DIVIDEND 16             5.7 %
   IN      42                                   131 DAYS
  ØUT      42                                   = 15.8 %
 ØPTIØN    36                                   ANNUALLY
MARGIN     77                                   BRK-EVEN
       -----                 -----              IS
TØTALS 7971                   8140              $ 38
MØRE? (1=YES)
?1
```

Example 1. Using Polaroid April 35's

```
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?102,.08
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?200,38.87,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?4,40,287
CØVERED WRITE NØ 102   12   10   76
BUY 200   SHARES AT 38.87
SELL 2     4   40 'S AT $ 287 EACH

        CØST                 PRØCEEDS              RETURN
YØUR $ 3887             STØCK   8000            $ 609 ØN
LØAN $ 3887             ØPTIØN   574             3504 IS
BRØKER'S                DIVIDEND 16              17.3 %
   IN      42                                    131 DAYS
  ØUT      42                                    = 48.2 %
 ØPTIØN    32                                   ANNUALLY
MARGIN     91                                   BRK-EVEN
       -----                 -----               IS
TØTALS 7981                   8590               $ 35.8
MØRE? (1=YES)
?1
```

Example 2. Using Polaroid April 40's

```
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?103,.08
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?200,38.87,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?4,45,137

CØVERED WRITE NØ 103   12   10   76
BUY 200   SHARES AT 38.87
SELL 2     4   45 'S AT $ 137 EACH

        CØST                 PRØCEEDS              RETURN
YØUR $ 3887             STØCK  9000             $ 1304 ØN
LØAN $ 3887             ØPTIØN  274              3809 IS
BRØKER'S                DIVIDEND 16              34.2 %
   IN      42                                    131 DAYS
  ØUT      42                                    = 95.2 %
 ØPTIØN    29                                    ANNUALLY
MARGIN     99                                    BRK-EVEN
       -----                 -----               IS
TØTALS 7986                   9290               $ 32.3
MØRE? (1=YES)
?1
```

Example 3. Using Polaroid April 45's

INPUT	SOURCE
MM.DD.YY	This is the month, day, and year of the quotations.
MARGIN? (1=YES)	If the stock is bought on ***margin***, enter a 1, otherwise enter any number.
LOAN RATE (#.##)	This is the ***margin*** interest rate, expressed as a percentage. It is usually ½% above the prime lending rate. Ask your broker or use 7.50 as a good approximation.
OUTPUT TO BE 32 OR 64 CHARACTERS	The output is 16 lines high, and can be either 32 or 64 characters wide.
EVALUATION NO.	Your choice. If the program could handle strings, it would be the stock name.
DIVIDEND ($/SHARE/QTR.)	This information is available from many sources, including Barron's. It can be zero.
NO. SHARES	The number of shares purchased. It must be in multiples of 100 shares.
PRICE/SHARE	Fractional prices must be converted to decimal prices, for example 37¼ is 37.25. This information will be the last item in each line of the option quotations.
MARKET (1=NYSE,2=OTC)	Ask your broker if the stock can be purchased in the 3rd (OTC) market.
EXPIRATION MONTH	This is the number of the expiration month, 1=January, 4=April, etc. The program assumes all options expire on the 20th day.
STRIKE	On your option quotations, this is the number next to the month. For example, a "Jan35" is an option which expires in January and has a strike of $35/share.
COST/OPTION	Option prices are given on a per share basis. To compute the total cost per option, multiply the price by 100. For example, 2⅜ is $237.50 per option. Use the last or closing prices as input.

```
RUN
INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)

?12,10,76,1,7.5
ØUTPUT TØ BE 32 ØR 64 CHARACTERS?
?32
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)
?104,.08
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)
?200,35,0
??0
EXCHANGE IS 1 ØR 2 0
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)
?200,38.87,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN
?1,35,425
CØVERED WRITE NØ 104   12   10   76
BUY 200   SHARES AT 38.87
SELL 2     1   35 'S AT $ 425 EACH
     CØST        PRØCEEDS      RETURN
 YØUR $ 3887  STØCK   7000  $-68 ØN
 LØAN $ 3887  ØPTIØN   850   3181 IS
 BRØKER'S      DIVIDEND0    -2.1 %
     IN     42                41 DAYS
    ØUT     42                =-18.6 %
   ØPTIØN   34                ANNUALLY
 MARGIN     26                BRK-EVEN
        -----          -----  IS
 TØTALS 7918          7850    $ 39.2
MØRE? (1=YES)
?1
```

Example 4. Using Polaroid January 35's

```
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?105,.08
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?200,38.87,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?1,40,144

CØVERED WRITE NØ 105   12   10   76
BUY 200   SHARES AT 38.87
SELL 2     1   40 'S AT $ 144 EACH

    CØST        PRØCEEDS      RETURN
YØUR $ 3887 STØCK   8000  $ 369 ØN
LØAN $ 3887 ØPTIØN   288   3744 IS
BRØKER'S    DIVIDEND0     9.8 %
    IN     42                41 DAYS
   ØUT     42                = 87.2 %
  ØPTIØN   30                ANNUALLY
MARGIN     31                BRK-EVEN
        -----          -----  IS
TØTALS 7919          8288    $ 37
MØRE? (1=YES)
?0


STØP IN LINE 145
```

Example 5. Using Polaroid January 40's

The editing of data is extensive. Any item which is used in the computation is checked for reasonableness. If multiple errors exist in an input line, each one is described briefly along with the erroneous value before the datum is to be reinput.

By studying the examples given with the actual data, it will be apparent that there is a considerable difference between the rate of return for various strike prices, even with the same expiration date. This difference depends on whether the option is in, at, or out of the money. The option you choose depends upon your investment goals, your assessment of the market in general, and your stock during the life of the option. Figure 5 below illustrates this relationship.

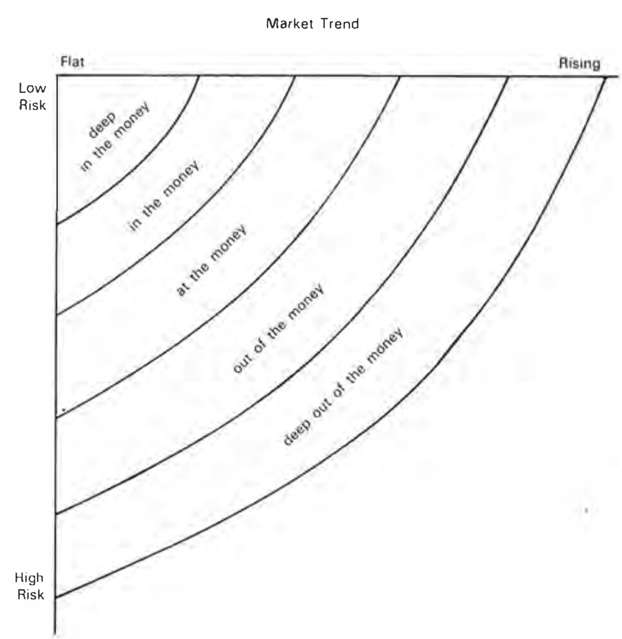


Figure 5. Market Trend vs Risk

When available a deep "in the money" ***call*** would be two or more strikes below the "at the money" ***call***. A deep "out of the money" ***call*** would be two or more strikes above the "at the money" ***call***.

GETTING STARTED

If you are ready for option writing, I recommend that you do some further reading. Three excellent books available on option strategies are listed at the end of this article. They also include sections on income tax requirements, a subject certainly worth studying. Choose a broker who specializes in options, who understands ***hedges***, order placement, floor execution, and most importantly, your investment goals and will recommend suitable strategies in tune with these goals. Don't settle for less, remember it's your money!

A special thanks to Ward Christenson for the use of his ALTAIR 8800 system and assistance during one marathon session of inputting, debugging and testing.

PROGRAM SPECIFICATIONS AND REMARKS

The program is written in Processor Tech 5K Basic. Because of extensive comments and editing, its size is about 4.2K. Eliminating the comments and making the edit error descriptions more cryptic could reduce it to about 3K.

SUGGESTED READING

The New Options Market by Max G. Ansbacher (Walker & Co.). A good book for the beginning student of options. He covers strategies for buying options, naked writing, covered writing as well as more advanced strategies using options.

2. ***The Stock Options Manual*** by Gary L. Gastineau (McGraw-Hill). Another good book for the student of options. He introduces some probability theory into evaluation of options as well as graphically displaying various ***hedging*** strategies.
3. ***How the Experts Beat the Market*** by Thomas Noddings (Dow Jones-Irwin). Not for the beginner in options. He presents numerous ***hedging*** strategies using convertible bonds, warrants, and options.

DISCLAIMER (MY *HEDGE*)

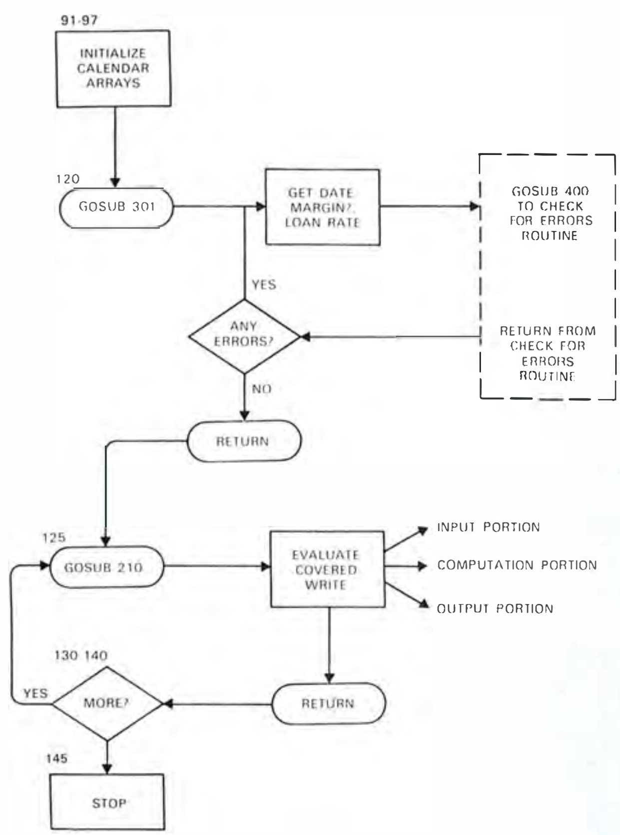
The charts, tables, and examples presented in this article are for illustration only, and should not be construed as recommendations for purchase and/or sale of any individual stocks or related options. Further, it is not intended to indicate nor imply in any manner that the methods described in this article can guarantee profitable results in the future.

EXPLANATION OF PROGRAM

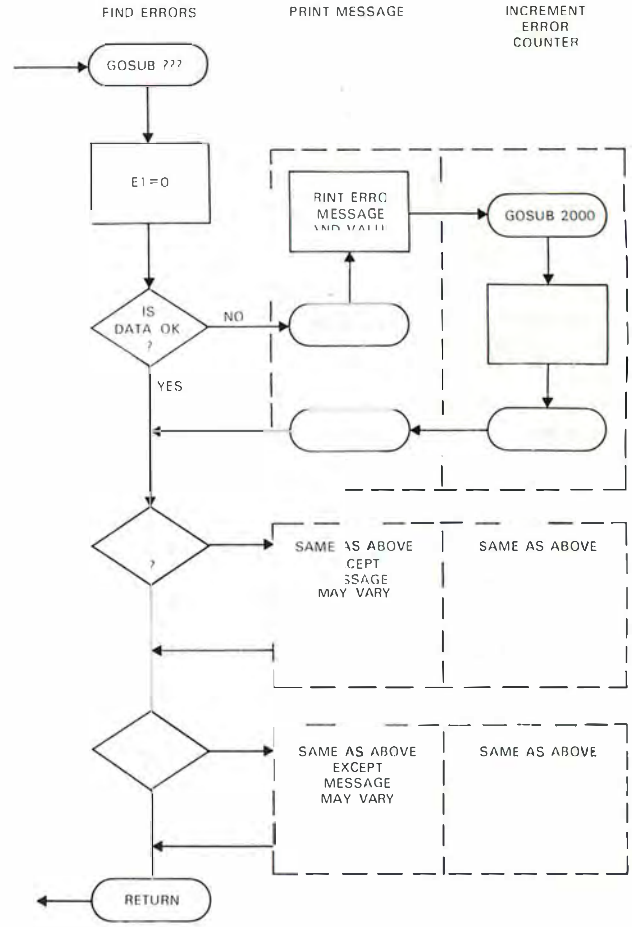
The program is highly modularized in design, that is, each function is processed by a separate, independent group of statements. By using this method, the basic program can be set up and expanded a piece at a time. Conversely, whole modules could be "lifted" and used without modification.

There are several features worth noting. First, liberal use of comments will help clarify the purpose of a module. Second, the error checking routines are extensive. In addition, the reason for the error or the allowable range of the value is displayed as well as the erroneous value. Third, the style of programming is made as consistent as possible. For example, extensive use of the GOSUB-RETURN statements allow the main program logic to remain in sequence. When a task must be done, the program GOSUB's to the task and RETURN's when it is done.

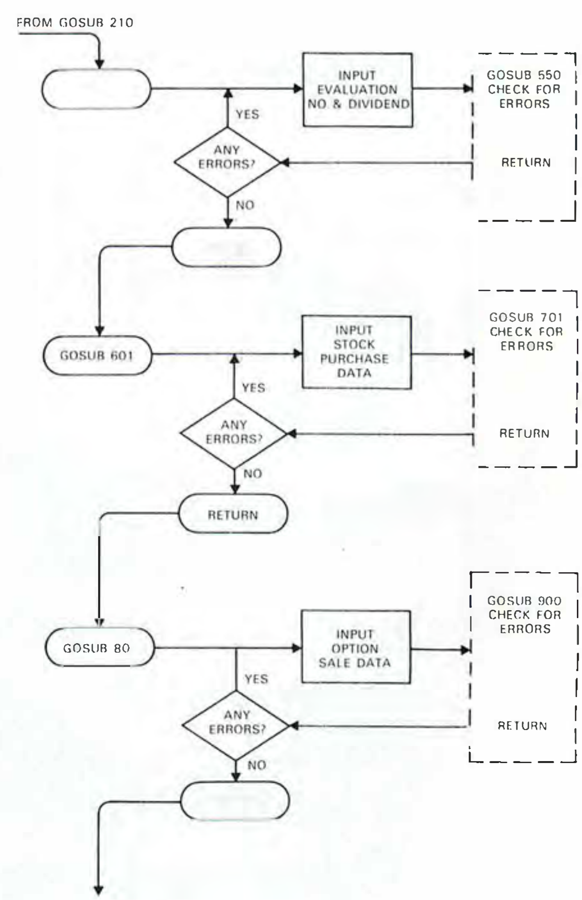
Fourth, GOTO statements were not used. In a few cases, the IF - THEN (go to) is used to move program control to reinitiate an input request or branch to a return statement.



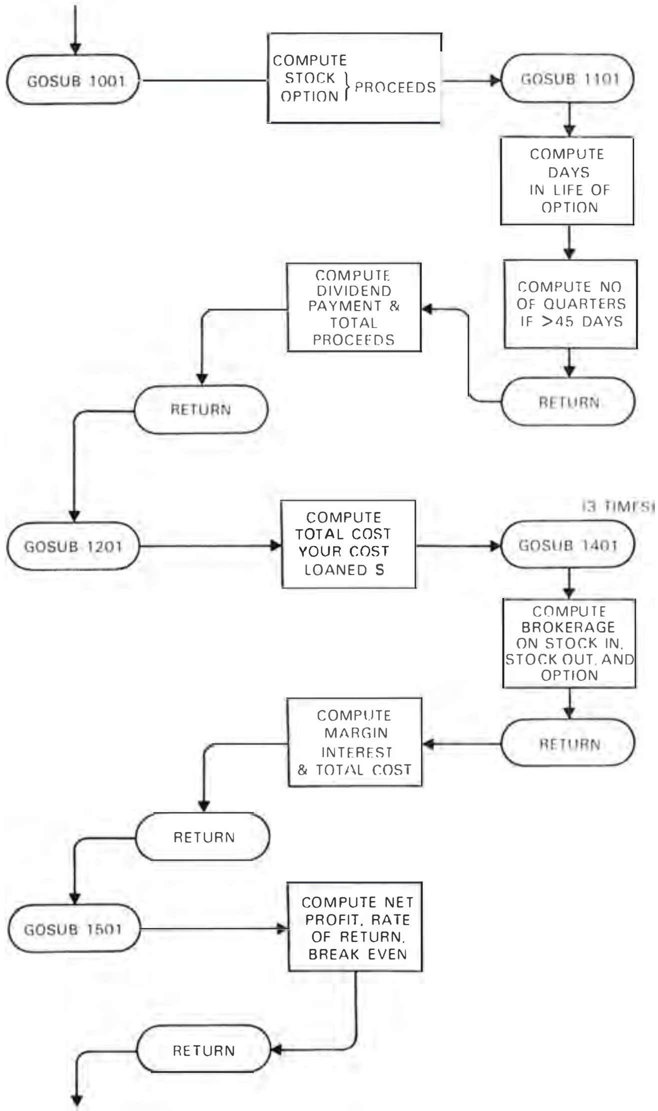
Main Program — Overview **Figure A**



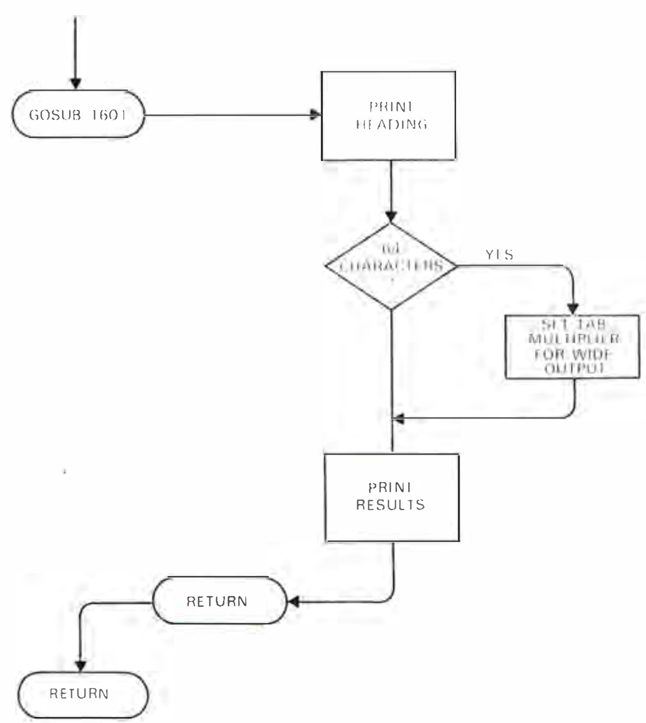
Check for Errors Routine **Figure B**



Evaluate Covered Write-Input Portions **Figure C**



Evaluate Covered Write-Computation **Figure D**



Evaluate Covered Write-Output **Figure E**

Example. Demonstration of Extensive Edits

```
RUN

INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)

?76,0,0,4,3.70
DATE ERRØR  76 0 0
DATE ERRØR  76 0 0
INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)

?76,0,0,1,3.0
DATE ERRØR  76 0 0
DATE ERRØR  76 0 0
RANGE=5 TØ 14%  3
INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)

?12,10,76,1,7.5
ØUTPUT TØ BE 32 ØR 64 CHARACTERS?
?32
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?100,6
DIVIDEND RANGE IS 0 TØ $5  6
EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)

?100,.08
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?125,38.87,3
NØT EVEN 100'S  125
EXCHANGE IS 1 ØR 2  3
NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)

?200,38.87,2
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?0,0,0
MØNTH ERRØR 0
STRIKE <10 ØR >3000
PRICE < $25 0
EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN

?4,40,287


CØVERED WRITE NØ 100   12   10   76
BUY 200  SHARES AT 38.87
SELL 2    4  40 'S AT $ 287 EACH

    CØST      PRØCEEDS    RETURN
YØUR $ 3887 STØCK  8000 $ 609 ØN
LØAN $ 3887 ØPTIØN  574  3504 IS
BRØKER'S    DIVIDEND 16  17.3 %
   IN     42             131 DAYS
  ØUT     42             = 48.2 %
 ØPTIØN   32             ANNUALLY
MARGIN    91             BRK-EVEN
      -----        -----  IS
TØTALS 7981        8590   $ 35.8
MØRE? (1=YES)
?0


STØP IN LINE 145

READY


10  REM STØCK MARKET HEDGE - CØVERED WRITING STRATEGY
20  REM WRITTEN BY EDWARD CHRISTIANSØN 12/15/76
90 DIM C(12)  ; REM SET UP JULIAN DATE ARRAY'
91 DATA 0,31,59,90,120,151,182,212,243,273,304,334
92 FØR I=1 TØ 12; READ C(I) ; NEXT I
95 DIM M(12)  ; REM SET UP MØNTH DESCRIPTIØNS'
97 FØR I=1 TØ 12; M(I)=I; NEXT I
120 GØSUB 301; REM SET UP INITIAL VALUES'
125 GØSUB 210; REM GET INPUT DATA AND EVALUATE PØSITIØN'
130 PRINT"MØRE? (1=YES)",
135 INPUT A1
140 IF A1=1 THEN 125
145 STØP
210 GØSUB 501; REM GET STØCK DESCRIPTIØN AND DIVIDENDS'
220 GØSUB 601; REM GET STØCK PURCHASE DATA'
230 GØSUB 801; REM GET ØPTIØN SALE SIDE DATA'
240 GØSUB 1001; REM CØMPUTE TØTAL PRØCEEDS IF CALLED'
250 GØSUB 1201; REM CØMPUTE TØTAL CØSTS'
260 GØSUB 1501; REM CØMPUTE NET PRØFITS, RATE ØF RETURN'
270 GØSUB 1601; REM DISPLAY RESULTS'
280 RETURN
301 PRINT"INPUT MM,DD,YY, MARGIN? (1=YES), LØAN RATE (#.##)"
```

```
305 INPUT M,D,Y,M3,'R
306 M1=1
310 GØSUB 400; REM CHECK FØR ØBVIØUS INPUT ERRØRS
315 IF E1 > 0 THEN 301
320 PRINT"ØUTPUT TØ BE 32 ØR 64 CHARACTERS? ",
325 INPUT W1
330 RETURN
400 E1 = 0; REM E1 IS AN ERRØR FLAG'
403 IF M<1 THEN GØSUB 450
406 IF M>12 THEN GØSUB 450
409 IF D<1 THEN GØSUB 450
412 IF D>31 THEN GØSUB 450
415 IF Y<0 THEN GØSUB 450
418 IF Y>99 THEN GØSUB 450
421 IF M3<>1 THEN 440; REM NØT MARGINED'
424 M1=.5; REM SET TØ 50% RATE'
427 IF R<5 THEN GØSUB 455
430 IF R>14 THEN GØSUB 455
440 RETURN
450 PRINT"DATE ERRØR ",M,D,Y
451 GØSUB 2000
452 RETURN
455 PRINT"RANGE=5 TØ 14% ",R
456 GØSUB 2000
457 RETURN
501 PRINT"EVALUATIØN NØ, DIVIDEND ($/SHARE/QTR)"
505 INPUT N9,D1
510 GØSUB 550; REM CHECK FØR ØBVIØUS ERRØRS'
515 IF E1 > 0 THEN 501
520 RETURN
550 E1 = 0
551 IF D1<0 THEN GØSUB 575
554 IF D1>5 THEN GØSUB 575
560 RETURN
575 PRINT"DIVIDEND RANGE IS 0 TØ $5 ",D1
576 GØSUB 2000
577 RETURN
601 PRINT"NØ. SHARES,PRICE/SHARE,MARKET(1=NYSE,2=ØTC)"
605 INPUT S1,P1,M9
610 GØSUB 701; REM CHECK FØR ERRØRS'
615 IF E1 > 0 THEN 601
620 RETURN
701 E1 = 0; REM EDIT STØCK DATA'
706 IF S1<100 THEN GØSUB 751
709 S9=INT(S1/100)
712 IF S1<>S9*100 THEN GØSUB 751
715 IF P1<10 THEN GØSUB 755
718 IF P1>300 THEN GØSUB 755
721 M9=INT(M9)
724 IF M9<1 THEN GØSUB 760
727 IF M9>2 THEN GØSUB 760
730 RETURN
751 PRINT"NØT EVEN 100'S ",S1
752 GØSUB 2000
753 RETURN
755 PRINT"PRICE<10 ØR >300 ",P1
756 GØSUB 2000
757 RETURN
760 PRINT"EXCHANGE IS 1 ØR 2 ",M9
761 GØSUB 2000
762 RETURN
801 PRINT"EXPIRATIØN MØNTH, STRIKE, CØST/ØPTIØN"
805 INPUT M6,S6,P6
810 GØSUB 900; REM CHECK FØR ERRØRS'
815 IF E1>0 THEN 801
820 RETURN
900 E1=0
903 Ø6=S1/100; REM CØMPUTE NØ.ØPTIØNS'
906 IF M6<1 THEN GØSUB 955
909 IF M6>12 THEN GØSUB 955
912 IF S6<10 THEN GØSUB 960
915 IF S6>300 THEN GØSUB 960
918 IF P6<25 THEN GØSUB 965
930 RETURN
955 PRINT"MØNTH ERRØR ",M6
956 GØSUB 2000
957 RETURN
960 PRINT"STRIKE <10 ØR >300",S6
961 GØSUB 2000
962 RETURN
965 PRINT"PRICE < $25 ",P6
966 GØSUB 2000
967 RETURN
1001 T1=INT(Ø6*100*S6); REM CØMPUTE PRØCEEDS IF CALLED'
1005 T2=INT(Ø6*P6); REM CØMPUTE ØPTIØN PRØCEEDS'
1010 GØSUB 1101; REM CØMPUTE DAYS IN LIFE AND # ØF QTRS'
1015 T3=INT(D1*S1*Q);REM TØTAL DIVIDEND INCØME'
1020 T4=T1+T2+T3; REM CØMPUTE TØTAL INCØME'
1025 RETURN
1101 Y1=Y*365; REM CØMPUTE DAYS IN LIFE'
1105 D2=C(M)+D+Y1
1110 D3 =C(M6)+20+Y1; REM ASSUME ØPTIØN EXPIRES ØN 20TH DAY'
1115 D4=D3-D2
1120 IF D4<0 THEN D4=D4+365
1125 Q=INT((D4/90)+.5); REM CØMPUTE QUARTERS FØR DIVIDENDS'
1130 RETURN
1201 C1=INT(S1*P1);REM CØMPUTE TØTAL CØST'
1205 C2=INT(C1*M1);REM C2 IS ØWN CAPITAL'
1210 C3=C1-C2
1215 M8=M9; REM M8 IS MARKET TYPE FØR BRØKERAGE CALCULATIØN'
1220 X=C1
1225 X1=S1/100
1230 GØSUB 1401; REM CØMPUTE BRØKERAGE ØN SHARES'
1235 B1=B; REM B IS RETURNED VALUE ØF BRØKERAGE FEE'
1240 X=T1
1245 X1=S1/100
1250 GØSUB 1401; REM CØMPUTE BRØKERAGE ØN SHARE WHEN CALLED'
1255 B2=B
1260 X=T2
1265 X1=Ø6
1270 M8=3; REM M8=3 MEANS CBØE'
1275 GØSUB 1401; REM CØMPUTE BRØKERAGE ØN ØPTIØN SALE'
1280 B3=B
1285 C4=C3-T2-T3+B1+B2+B3; REM C4 IS NET CAPITAL BØRRØWED'
1290 M2=INT(((R/100)*C4)*D4/365); REM MARGIN INTEREST CØST'
1291 IF M1=1 THEN M2=0; REM NØ INTEREST'
1292 C5=C2+C3+B1+B2+B3+M2; REM C5 IS TØTAL CØST'
1295 RETURN
1401 B=((X*.009)+22)+(X1*6); REM ØLD BRØKERAGE FEES'
1405 IF M8=1 THEN B=INT(B*.7); REM NYSE DISCØUNT'
1410 IF M8=2 THEN B=X1*21; REM ØTC CØST'
1415 IF M8=3 THEN B=INT(B*.82); REM CBØE'
1420 IF B<25 THEN B=25; REM MINIMUM BRØKERAGE'
1425 RETURN
1501 N1=T4-C5; REM CØMPUTE PRØFIT AND RATE ØF RETURN'
1505 C6=C2-T2-T3+B1+B2+B3+M2; REM C6 IS NET CAPITAL'
1510 R1=INT((N1*1000)/C6)/10
1515 R2=INT(3650*R1/D4)/10
1520 R3=(INT((C1-N1)*100)/S1)/100; REM LØWER BREAKEVEN'
1525 RETURN
1601 PRINT"CØVERED WRITE NØ",N9,M,D,Y
1605 PRINT"BUY",S1," SHARES AT",P1
1610 PRINT"SELL",Ø6,"  ",M(M6),S6,"'S AT $",P6,"EACH"
1611 PRINT
1614 W2=1
1615 IF W1>60 THEN W2=1.9
1616 PRINT TAB(4*W2),"CØST",TAB(13*W2),"PRØCEEDS",
1617 PRINT TAB(W2*24),"RETURN"
1618 PRINT"YØUR $",C2,TAB(W2*12),
1620 PRINT"STØCK ",T1,TAB(W2*23),"$",
1622 PRINTN1,"ØN"
1624 PRINT"LØAN $",C3,TAB(W2*12),
1626 PRINT"ØPTIØN ",T2,TAB(W2*23),
1628 PRINTINT(C6),"IS"
1630 PRINT"BRØKER'S",TAB(W2*12),
1632 PRINT"DIVIDEND",T3,TAB(W2*23),%Z1%,R1,"%",%%
1634 PRINT"   IN   ",B1,TAB(W2*23),
1636 PRINTD4,"DAYS"
1638 PRINT"  ØUT   ",B2,TAB(W2*23),
1640 PRINT" =",R2,"%"
1642 PRINT" ØPTIØN ",B3,TAB(W2*23),
1644 PRINT" ANNUALLY"
1646 PRINT"MARGIN  ",M2,TAB(W2*23),
1648 PRINT" BRK-EVEN"
1649 PRINTTAB(6),"-----",TAB(W2*12),"       -----",TAB(W2*23),
1650 PRINT" IS"
1652 PRINT"TØTALS",C5,TAB(W2*12),"       ",T4,TAB(W2*23),
1654 PRINT" $",(INT(R3*10))/10
1660 RETURN
2000 E1=E1+1
2001 RETURN
```


CIRCLE INQUIRY NO. 9

MORE MERLIN MAGIC!!!

Now . . . the already tremendous value of MERLIN, the best, most versatile video interface, is an even greater value with the introduction of two inexpensive ***add-on*** features.

SUPER DENSE GRAPHICS
$39.00
320H X 200V RESOLUTION

MERLIN's Super Dense Graphics is an add-on board which mounts directly to the MERLIN Video Interface. Each of 64,000 graphic dots is ***individually*** controllable and located ***on-screen.*** The kit includes a manual with X-Y coordinate transformation software listing.

SOFTWARE CASSETTE I/O
$29.00
1500 BAUD

MERLIN's Software ***1500 Baud*** Cassette I/O is an external add-on to the MERLIN Video Interface. It connects, along with a keyboard, to the MERLIN KYBD and I/O Port through a 16 conductor flat cable. The kit includes a manual with a listing of all necessary software, also available as part of the MEI ROM.

For more information, fast, write us direct.

Pricing:

MERLIN Assm.	$349.00
MBI ROM/RAM	$ 39.95
Super Dense Add-on	$ 39.00
MERLIN Manual	$ 10.00
MERLIN Kit	$269.00
MEI ROM	$ 34.95
Cassette I/O Add-on	$ 29.00

Master Charge and BankAmericard accepted.

MiniTerm Associates, inc.

Box 268, Bedford, Mass. 01730 (617) 648-1200

By Tom Mazur

WARP FACTORS

Captain's Log: Star Date 3814. 2: Quadrant 14-Z2 Sector A-11; in polar orbit around third planet, Tellian system...

"and all corrections have been made in the warp mechanisms at this remote site. Operations are satisfactory now. Fortunately, this planet is uninhabited — no telling how much havoc the warp forces could have caused to organic life."

"I close this entry with the narrative of a baffling discovery we made just before beaming up."

"Kkkhh? Archie? Where are you? Let's go."

"I believe Mr. Tyson is inspecting the null zone area behind the flux modulators, Captain."

"Aye, Captain; could you and Mr. Kkkhh come over here? I've found something."

"Okay, Archie, what is it? Some residual temporal perturbation still in the area?"

"No, Captain, nothing like that. See here, in the sand — some kind of apparatus that is not a part of the flux modulators, or any other system at this site."

"Let's have a look."

"Well, Mr. Kkkhh, any ideas?"

"Interesting, Captain. The printed text on this apparatus appears to be in an archaic form of English."

"H'mm, yes, I see what you mean. Educator two microcomputer HEP kit by Motorola, if I interpret correctly."

"Captain?"

"Yes, Mr. Tyson?"

"I've scanned engineering references in the ship's archives many times, but I've na' come across a heading called 'microcomputer.' "

"Perhaps, Mr. Tyson, you did not access the references in a thorough manner."

"Your ears! Mr. Kkkhh!"

"Gentlemen! Time to beam up. Bring the unit along, we'll have Computer do an analysis."

"Computer."

"Working."

"Identify the purpose, origin and age of this unit."

"Unit is a microcomputer, a primitive ancestor of myself. Point of origin, Earth circa late 1900s, Earth calendar. Age, less than thirty days old."

"What?!!"

"Easy, Archie. Computer: elaborate."

"Unit is the Educator II, a stored-program, general-purpose digital computer kit that was offered by the HEP/MRO department of an organization known as Motorola. A block diagram of the unit is contained in my memory banks."

"Computer, display the diagram; continue the description."

"Yes, Captain. The unit contains M6800 microcomputer products. Specifically, a HEP C4801L microprocessing unit (MPU), a C4821L Peripheral Interface Adapter (PIA) and a C4811L 128x8 Random Access Memory (RAM). There are provisions on the single board kit for another 128x8 RAM. A built-in interface allows additional user programs to be stored on and loaded from audio cassette tapes. User programs on cassettes may be accessed and loaded into RAM by means of a software search feature. Said interface and search features are routines contained in Programmable Read Only Memories (PROMs) on the board. The two 512x4 PROMs also contain an executive monitor for user/computer interaction by means of front panel switches and a light emitting diode (LED) display. A clock circuit, running at 0.5 MHz, is also on-board. The unit is self-contained; all parts are included as well as a complete construction manual. The unit requires a 5 V, 1.2 A power source."

"Computer, what other modes of communication were available to the users of this Educator II?"

"At the point in time when this unit was introduced, Mr. Kkkhh, tactile/visual interaction was in vogue. Verbal interaction was in its infancy."

"Highly interesting. Computer, how was this tactile/visual communication implemented?"

"User/computer dialogue was at the machine code level for the initial Educator II kit. Object code patterns were set by means of eight single pole, double throw switches (SPDT) and loaded, examined and modified by three SPDT momentary switches. Bytes of code were displayed on eight LEDs. Subsequent Motorola products and user system expansions produced highly articulate configurations."

"Computer, continue with the description of the Educator II.

"Yes, Captain. Edge connectors on the board provide an interface to the PIA as well as all address, data and control bus signals for system expansion. A

test-as-you-build approach provided for an accurate, low-error kit construction. Assembly time was typically one Earth evening. A complete construction manual as well as a full complement of parts helped to hold the assembly time to such short duration. Tutorial support documentation, designed as aids for understanding MPUs and programming concepts, was also included. Shortly after Educator II's introduction, other HEP support kits, such as a power supply and memory boards, were also introduced."

"All well and good, but, Captain, we still ha' one verrry perplexin' question: How can something from the 20th century, Earth, cross par-secs of space and still be new?"

"Yes, Archie, perplexing, true; but not entirely improbable, eh, Kkkhh?"

"Quite plausible, Captain."

"Computer, state the particulars of Educator II's introduction."

"Educator II Microcomputer Kit introduced March 1977, Earth calendar. Place of manufacture: northern hemisphere of Earth, in the country then known as the United States of America, in the State then known as Arizona. When this culture reached technological Level 8, this manufacturer identified as Motorola, Inc. maintained a lead in the design and application of signal technology."

"Does that suggest anything to you, Mr. Kkkhh?"

"The factors are favorable . . . simplicity, economy, availability. Yes, given such a potential target area, the probability of exchange would indeed be high. There is, of course, only one way to enforce this hypothesis, Captain."

"Do the honors, Mr. Kkkhh."

"Computer."

"Working."

"Computer, state the occurrence and periods of spatial alignments of Earth and Tellian III between January, 1977, Earth calendar, and Star Date 3814.2. Specify Star Date."

"Occurrence: 1977.5; 2995.84; 3814.13; Period: 35 Star Hours aligned, 140 Star Hours non-aligned. Fifty-two pulses per occurrence."

"H'mmm; yes, highly probable, Captain. An exchange could occur again at a conjugate site."

"Agreed Archie, take this Educator II and beam it down to the location that is diametrically opposite to the location at which you found it."

"Uh, aye, Captain. I take it then that you don't want to keep this Educator II in ships stores?"

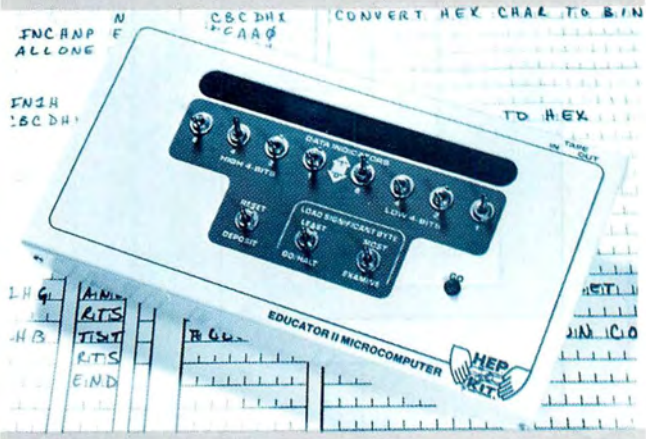
"Right, Archie. No matter how this microcomputer appeared on Tellian III, we owe the original owner the right to retrieve it. Who knows but that this, this Educator II may be the seed bed of ideas that led to our computer."

"Aye, Captain."

"Mr. Ooll, lay in a course to Digitus IV, warp factor five."

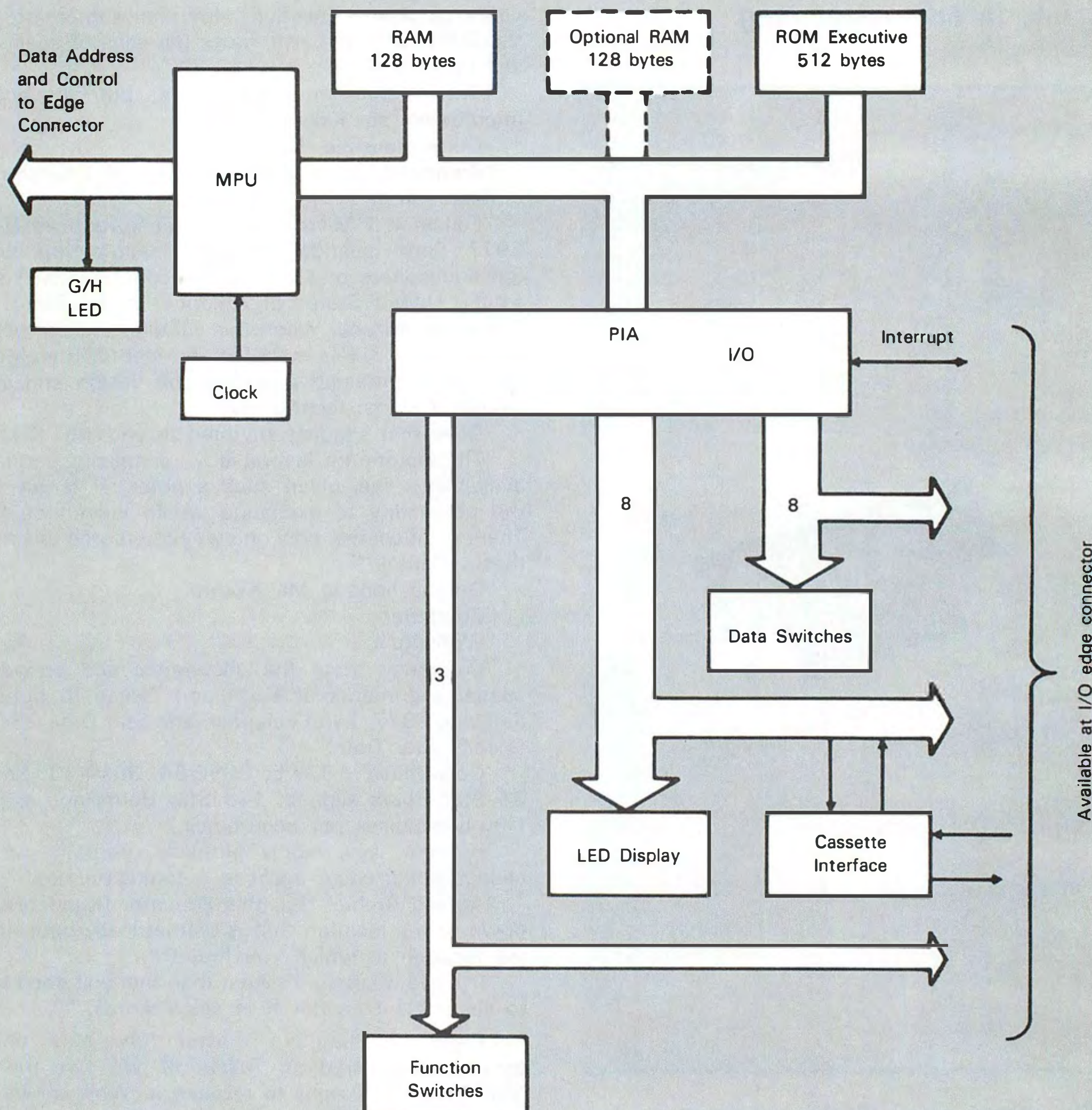
"Aye, aye, Captain."

At technology Level 8 great strides were made in microcomputing



Educator II Microcomputer

Do those aboard the ancient vessels in Space know what they contributed?



HEP Educator II Systems Block Diagram

Apple Introduces the First Low Cost Microcomputer System with a Video Terminal and 8K Bytes of RAM on a Single PC Card.

The Apple Computer. A truly complete microcomputer system on a single PC board. Based on the MOS Technology 6502 microprocessor, the Apple also has a built-in video terminal and sockets for 8K bytes of onboard RAM memory. With the addition of a keyboard and video monitor, you'll have an extremely powerful computer system that can be used for anything from developing programs to playing games or running BASIC.

Combining the computer, video terminal and dynamic memory on a single board has resulted in a large reduction in chip count, which means more reliability and lowered cost. Since the Apple comes fully assembled, tested & burned-in and has a complete power supply on-board, initial set-up is essentially "hassle free" and you can be running within minutes. At $666.66 (including 4K bytes RAM!) it opens many new possibilities for users and systems manufacturers.

You Don't Need an Expensive Teletype.

Using the built-in video terminal and keyboard interface, you avoid all the expense, noise and maintenance associated with a teletype. And the Apple video terminal is six times faster than a teletype, which means more throughput and less waiting. The Apple connects directly to a video monitor (or home TV with an inexpensive RF modulator) and displays 960 easy to read characters in 24 rows of 40 characters per line with automatic scrolling. The video display section contains its own 1K bytes of memory, so all the RAM memory is available for user programs. And the Keyboard Interface lets you use almost any ASCII-encoded keyboard.

The Apple Computer makes it possible for many people with limited budgets to step up to a video terminal as an I/O device for their computer.

No More Switches, No More Lights.

Compared to switches and LED's, a video terminal can display vast amounts of information simultaneously. The Apple video terminal can display the contents of 192 memory locations at once on the screen. And the firmware in PROMS enables you to enter, display and debug programs (all in hex) from the keyboard, rendering a front panel unnecessary. The firmware also allows your programs to print characters on the display, and since you'll be looking at letters and numbers instead of just LED's, the door is open to all kinds of alphanumeric software (i.e., Games and BASIC).

8K Bytes RAM in 16 Chips!

The Apple Computer uses the new 16-pin 4K dynamic memory chips. They are faster and take ¼ the space and power of even the low power 2102's (the memory chip that everyone else uses). That means 8K bytes in sixteen chips. It also means no more 28 amp power supplies.

The system is fully expandable to 65K via an edge connector which carries both the address and data busses, power supplies and all timing signals. All dynamic memory refreshing for both on and off-board memory is done automatically. Also, the Apple Computer can be upgraded to use the 16K chips when they become available. That's 32K bytes on-board RAM in 16 IC's—the equivalent of 256 2102's!

A Little Cassette Board That Works!

Unlike many other cassette boards on the marketplace, ours works every time. It plugs directly into the upright connector on the main board and stands only 2" tall. And since it is very fast (1500 bits per second), you can read or write 4K bytes in about 20 seconds. All timing is done in software, which results in crystal-controlled accuracy and uniformity from unit to unit.

Unlike some other cassette interfaces which require an expensive tape recorder, the Apple Cassette Interface works reliably with almost any audio-grade cassette recorder.

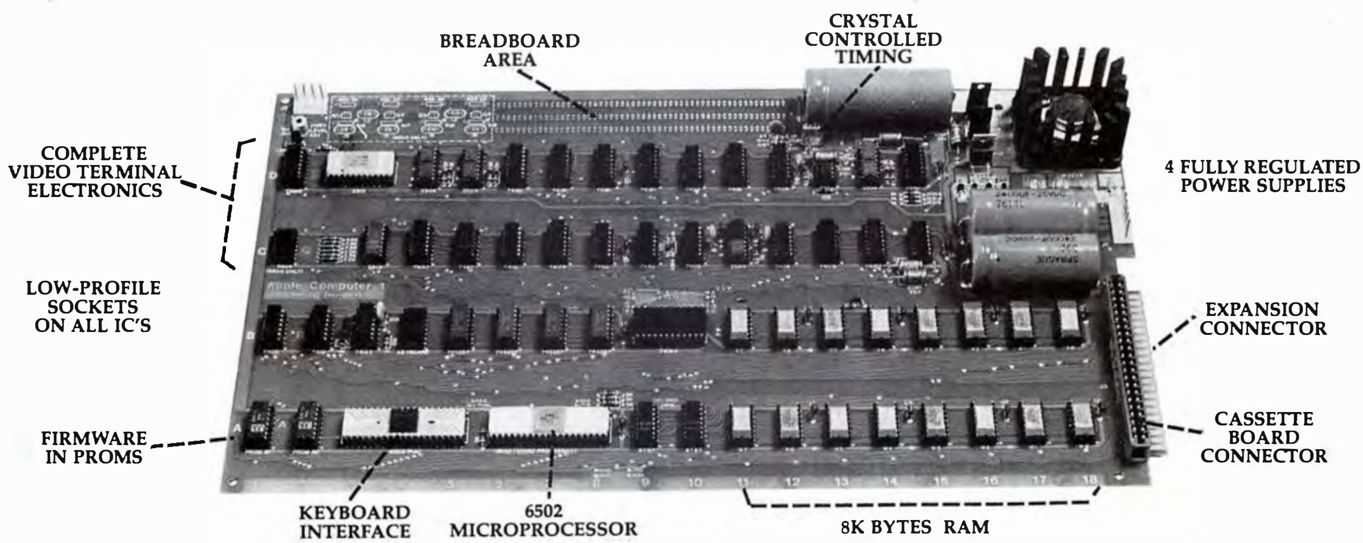
Software:

A tape of **APPLE BASIC** is included free with the Cassette Interface. Apple Basic features immediate error messages and fast execution, and lets you program in a higher level language immediately and without added cost. Also available **now** are a dis-assembler and many games, with many software packages, (including a macro assembler) in the works. And since our philosophy is to provide software for our machines free or at minimal cost, you won't be continually paying for access to this growing software library.

The Apple Computer is in stock at almost all major computer stores. (If your local computer store doesn't carry our products, encourage them or write us direct). **Dealer inquiries invited.**

Byte into an Apple $666.66*

* includes 4K bytes RAM



APPLE Computer Company • 770 Welch Rd., Palo Alto, CA 94304 • (415) 326-4248

FLIP OVER OUR FLOPPY

Only $750 from Peripheral Vision.

Peripheral Vision is a brand-new company that's dedicated to selling reasonably priced peripherals for various manufacturers' CPU's.

We think you'll flip over our first product.

It's a full-size floppy disk for the Altair-Imsai plug-in compatible S-100 BUS. And it's available for as low as $750. Here are the features:

- 1 interface card supports 4 drives
- Stores over 300,000 bytes per floppy
- Bootstrap EPROM included—no more toggling or paper tape
- Completely S-100 plug-in compatible
- Interface cabling included
- Drive is from Innovex (the originator of the floppy concept)—assembled and tested
- Interface card design is licensed from Dr. Kenneth Welles and the Digital Group
- Disk operating system with file management system included on floppy
- Cabinet and power supply optional

Prices:

	Kit	Assm.
Interface card kit and assembled and tested drive	$750	$850
Power supply— +24V at 2A	45	65
Cabinet—Optima, blue	—	85

Now, a little more about our company.

Peripheral Vision may be brand-new, but we have some old-fashioned ideas about how to run our business.

We know there are serious incompatibilities among the different manufacturers' peripherals and CPU's. We want to get them together. And, we want to bring significant new products to market—products consisting of everything from adaptation instructions/kits for hardware and software to major new products.

It's a tall order, but we feel we're up to the task. Peripheral Vision has already obtained a license from The Digital Group to adapt versions of some of their products to the S-100 BUS. And we're working on getting more from other companies.

Most important to our customers, Peripheral Vision is committed to helping you get along with your computer. We'll do all we can to make it easy.

Write us now for all the information on our company, our philosophy and our exciting line of products. And be prepared to flip over all of it.

PERIPHERAL VISION

P.O. Box 6267 / Denver, Colorado 80206 / (303) 733-1678

Send me the works, and I just might flip over it!

Name ______________________________

Address ______________________________

City/State/Zip ______________________________

BUILDING A 12-BIT ANALOG TO DIGITAL CONVERTER FOR REAL TIME PROBLEMS

By Roger W Brown

In order to make your computer aware of its environment it is necessary to input sensory data into its memory. Since most sensory data encountered in our environment are analog in nature they must be converted to digital value before the computer can understand them. This conversion process is accomplished by using an A/D converter as an interface between your computer and the analog signal supplied.

The heart of any real world interface is an analog to digital (A/D) converter. This A/D converter functions as a translator for the various analog stimuli from the outside world converting them into computer code. The analog stimuli are first converted into analog voltages by a transducer, such as a temperature module, and then input to the A/D converter which transforms the analog signal into its respective digital value. In order to minimize cost, additional sensor modules may be connected to a multiplexer board to allow several sensors to share the same A/D converter. Likewise, the output from the computer may be converted into its analog equivalent by a digital to analog (D/A) converter. This permits the computer to do such things as drawing plots, or displaying graphics on an oscilloscope or television screen, or generating a computer voice to talk to you. Many outputs may be generated at the same time by using one D/A converter and another multiplexer board, to reverse the above multiplexer procedure.

The A/D converter will perform the actual conversion of the analog signal into a binary word, for processing by the computer. The larger the binary word the greater the resolution and the more accurately the computer will be informed about the input signal. For example, a 10 bit A/D converter set for ± 5 volt operation has 512 steps of resolution between 0 and 5 volts and 1024 steps between -5 and $+5$ volts. This amounts to a resolution of better then 0.010 volts or 10mV per step. Consequently a 12 bit unit would have resolution to 2mV. The resolution of the A/D converter best suited to your needs will depend upon your particular application. However, it is well to remember that a 16 bit unit can be used as a 10 or 8 bit A/D converter when less resolution and greater speed are required, but an 8 or 10 bit unit cannot be used as a 16 bit A/D converter.

Once you have interfaced your computer with an A/D and/or a D/A converter module you are ready to hook-up the external sensors. If more than one sensor for an A/D converter or more than one output for a D/A converter are desired, a multiplexer must be interfaced between the converter module and the outside equipments.

Uses to which you could put your computer requiring A/D conversion are extremely numerous so only a few are given here. Your computer could be used around the house to provide you with a very sophisticated fire, smoke or intruder alarm system without the false alarm flaws of other systems presently on the market. Or it can be used to monitor outside weather conditions to forewarn of the pending dangers from water pipes and car radiators freeze up. As an external temperature monitor using wind speed and pressure gauges the system can predict local weather conditions for you. If you have a desire for privacy or need security you could use your computer to scramble your phone communications. In the garage your computer can monitor the operating systems of your car. The new Volkswagens have a connector on them specifically for this purpose. Auto systems such as timing, dwell, combustion, air pressure, engine torque, cam wear, plug and wire conditions, etc. can be monitored for preventative maintenance, tune-ups, and repairs. If you do much boating your computer can keep track of your speed, time, wind and water speed. From this it can tell you your position at any time and instruct you on course change to make to reach a desired target in the least amount of time, taking drift and winds into account. Tied to a radio receiver it can function as your inertial navigation system by receiving and processing Loran or Navy satellite signals. Onboard it could also serve as the processing center for side scan sonar, eliminating the costly recording and processing equipment now in use. By interfacing a microphone to your computer it could be taught to understand voice commands and with the addition of a speaker it could even talk to you. With the new MOS matrix IC's your computer could be given sight and used to monitor any room in your house. The list is endless and these are but a few uses shown here. However, to provide sight and sound functions, a very large memory would be required.

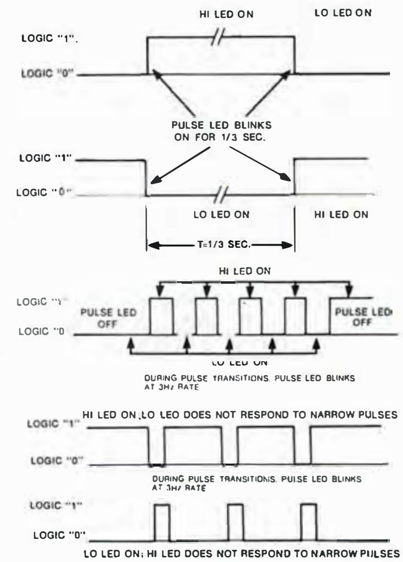
Logic Probe 1 is a compact, enormously versatile design, test and trouble-shooting tool for all types of digital applications. By simply connecting the clip leads to the circuit's power supply, setting a switch to the proper logic family and touching the probe tip to the node under test, you get an instant picture of circuit conditions.

LP-1's unique circuitry—which combines the functions of level detector, pulse detector, pulse stretcher and memory—makes one-shot, low-rep-rate, narrow pulses—nearly impossible to see, even with a fast scope—easily detectable and visible. HI LED indicates logic "1", LO LED, logic "0", and all pulse transitions—positive *and* negative as narrow as 50 nanoseconds—are stretched to ⅓ second and displayed on the PULSE LED.

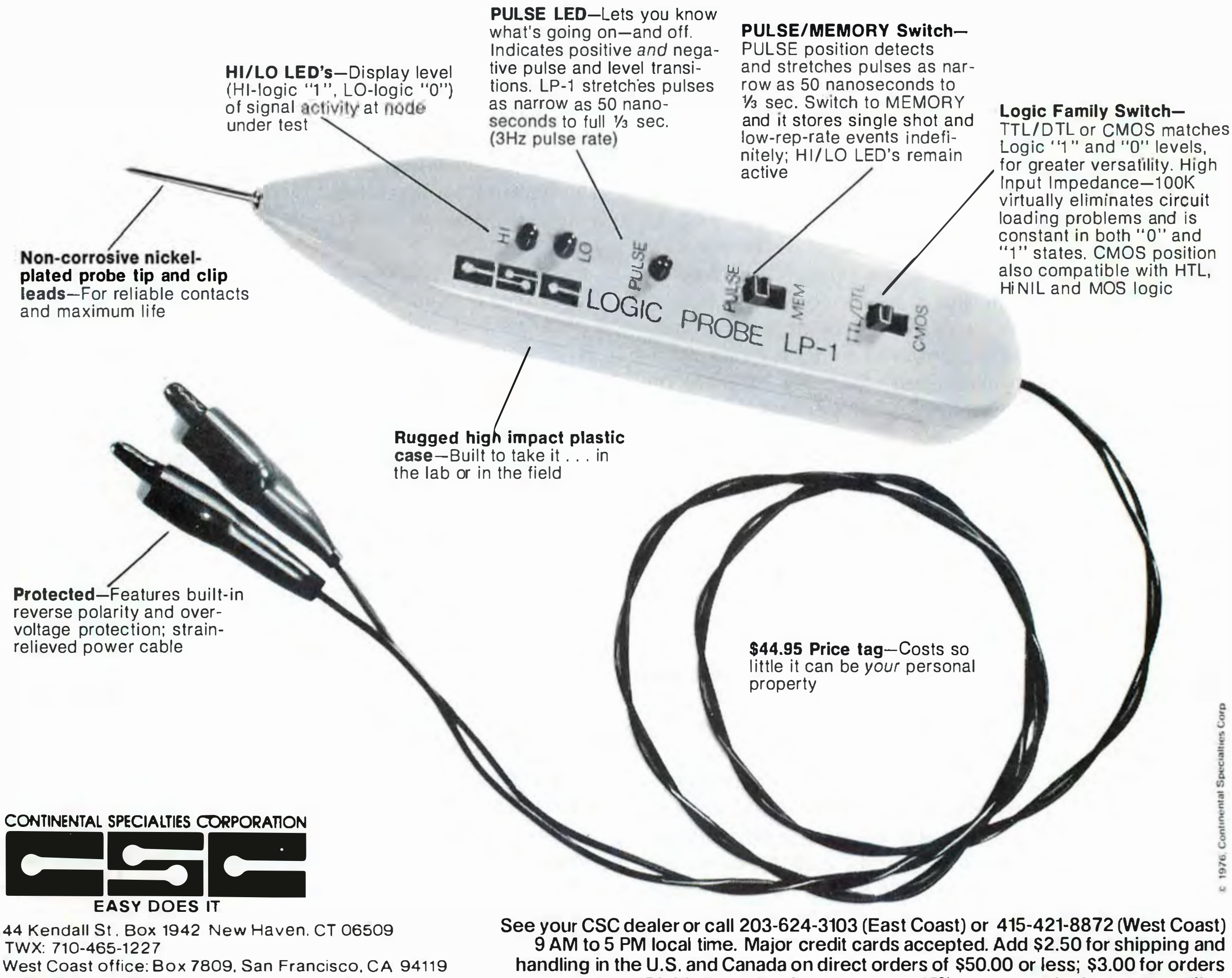
By setting the PULSE/MEMORY switch to MEMORY, single-shot events as well as low-rep-rate events can be stored indefinitely.

While high-frequency (5-10MHz) signals cause the "pulse" LED to blink at a 3Hz rate, there is an additional indication with unsymmetrical pulses: with duty cycles of less than 30%, the LO LED will light, while duty cycles over 70% will light the HI LED.

In all modes, high input impedance (100K) virtually eliminates loading problems, and impedance is constant for all states. LP-1 also features over-voltage and reverse-polarity protection. Housed in a rugged, high-impact plastic case with strain-relieved power cables, it's built to provide reliable day-in, day-out service for years to come.



CSC'S MULTI-FAMILY LOGIC PROBE 1. AT $44.95, IT DIGS UP A LOT OF INFORMATION WITHOUT BURYING YOUR BUDGET.



CONTINENTAL SPECIALTIES CORPORATION

CSC

EASY DOES IT

44 Kendall St., Box 1942 New Haven, CT 06509
TWX: 710-465-1227
West Coast office: Box 7809, San Francisco, CA 94119
TWX: 910-372-7992

See your CSC dealer or call 203-624-3103 (East Coast) or 415-421-8872 (West Coast) 9 AM to 5 PM local time. Major credit cards accepted. Add $2.50 for shipping and handling in the U.S. and Canada on direct orders of $50.00 or less; $3.00 for orders over $50.00. On all foreign orders add 15% to cover shipping and handling.

There are three basic types of A/D converters: ramp, successive approximation and tracking. First in the order mentioned is the ramp converter which uses a countup method of conversion and is best suited for very slow changing data. The major drawbacks to this type of converter are its extremely slow speed of 2 clock pulses per conversion and that data can only be taken out at the end of the conversion cycle. The second type of converter is the successive approximation A/D. This type is the most expensive as it requires additional data holding circuits; but then it is the fastest type of converter requiring only one clock cycle for each bit of resolution. The disadvantage associated with this type of converter other than high cost is its method of only sampling the input signal and then making a quick approximation of what it might be. Since the input signal is only sampled every N clock pulses this could amount to considerable error when observing very fast signals without a tell-tale error sign. The output from this type of converter is only true at the end of the conversion cycle. These types of converters are easily recognized as conversion time is always specified for the unit. The third type is the tracking A/D converter. This unit is a compromise between the first two types of converters. Essentially this type locks on to the input analog signal and as long as the slew rate of the loop is not exceeded the digital output will equal exactly the analog input to the resolution of the converter. If the slew rate is exceeded, the output immediately becomes erratic and warns you of the need for adjustment. What must be done to observe signals that are faster than the slew rate of the converter is to reduce the input amplitude and you are back in business again at a decreased resolution. An additional advantage of the tracking A/D converter over the other two types discussed here is the continuous availability of data at the output of the converter.

This project will discuss the construction of a 12-bit tracking A/D Converter. The entire schematic of the converter is shown in Figure 1 and the schematic for the power supply is shown in Figure 2.

ABOUT THE CIRCUIT:

An analog signal is presented to the input of IC4 (on an operational amplifier). This signal may be either a D.C. level or an A.C. signal. IC4 is used as in input buffer amplifier. With the addition of one potentiometer its gain may be made variable, or with two diodes, its gain may be made logarithmic. These circuit designs are available in application notes for the IC manufacturer. The output of IC4 drives the low impedance input of a comparator, IC5. C9 is used to reduce the effects of high frequency noise on the output of IC5, that may be present on the input signal. IC5 is in the A/D feedback loop which consists of ICs 6 through 14. When the feedback signal, on pin #3 of

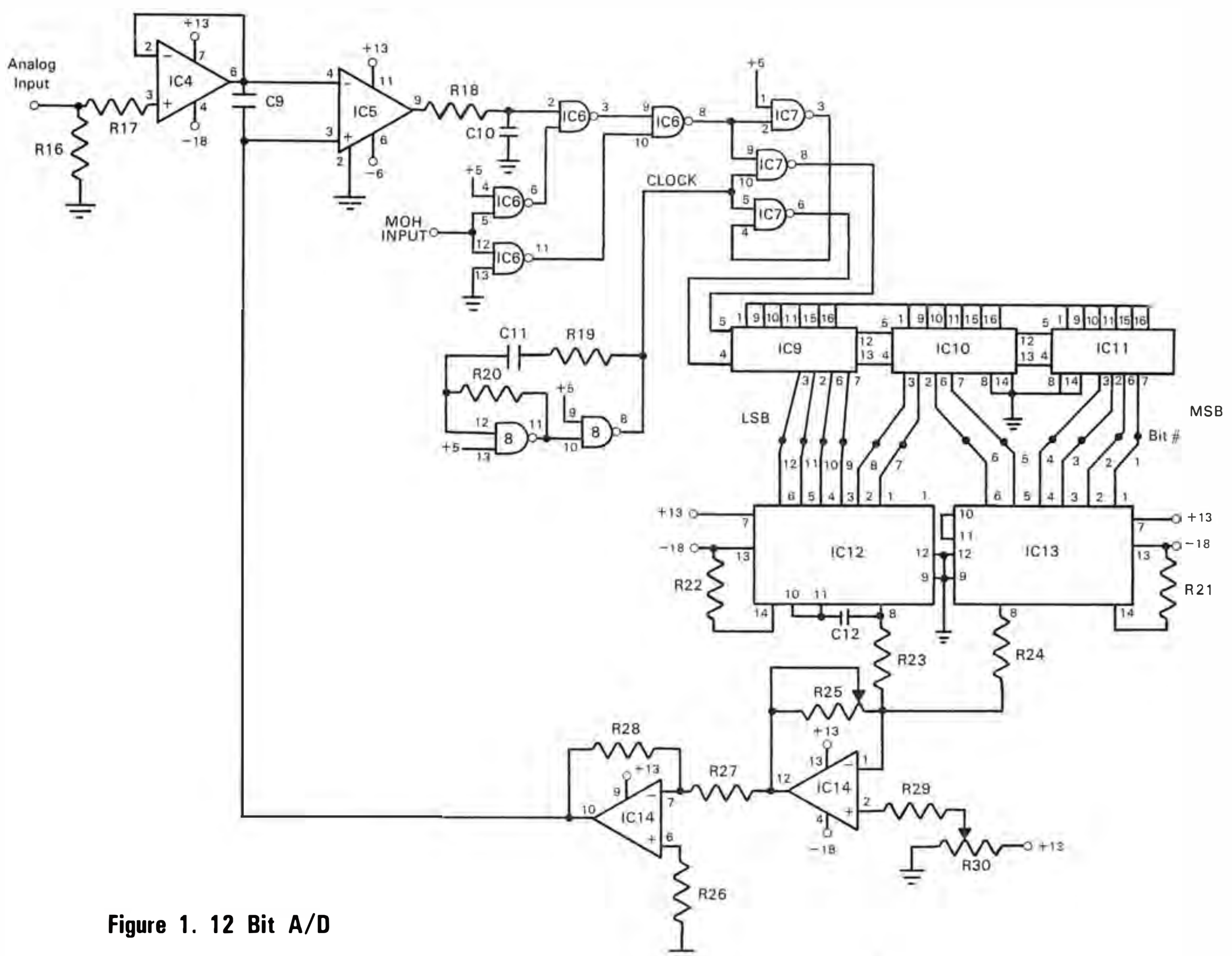


Figure 1. 12 Bit A/D

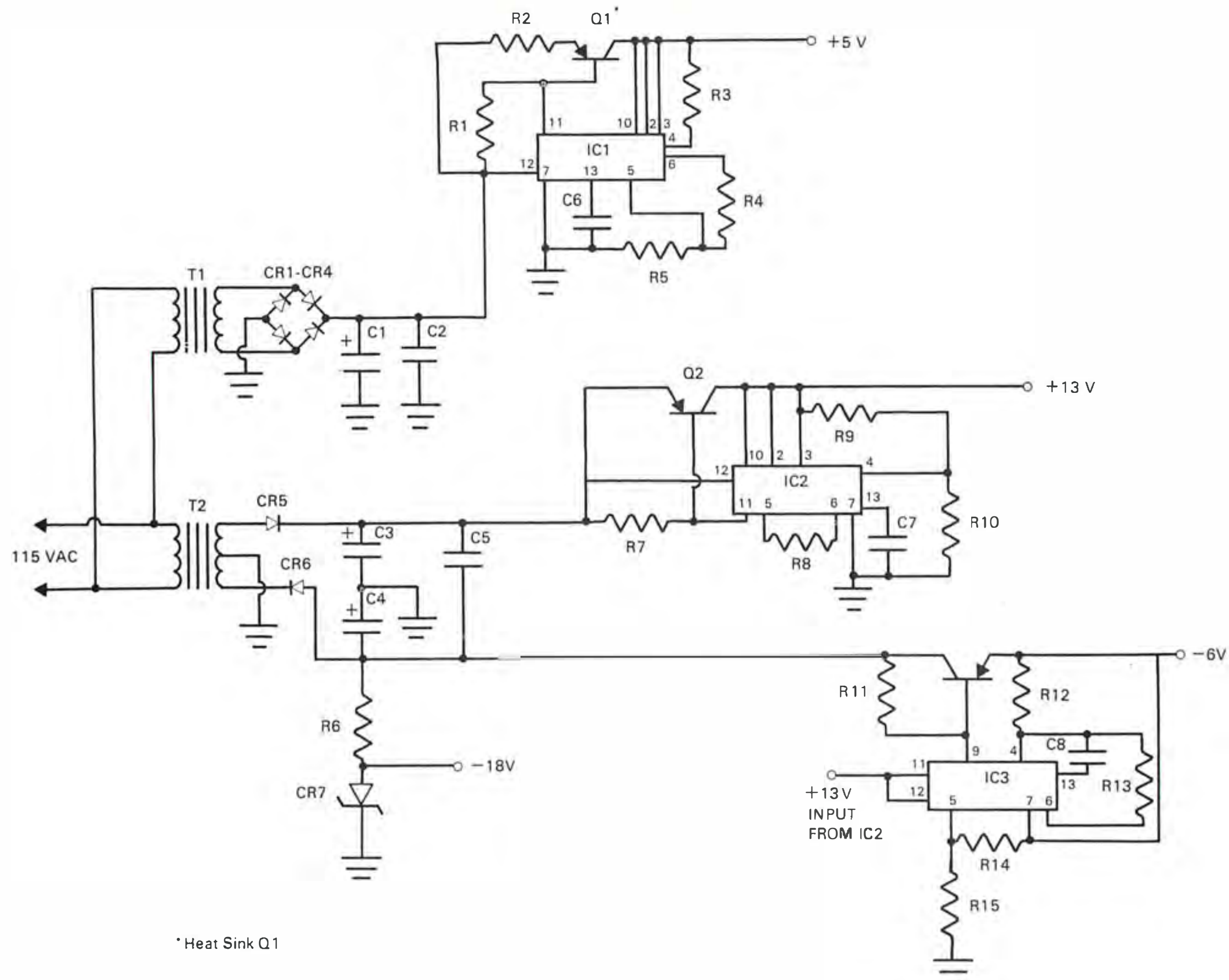


Figure 2. Power Supply

IC5, is greater than the true input signal, the output goes positive reversing the up/down counters, IC9 through IC11. The output of IC5 after filtering out the switching transients through R18 and C10, goes to the input of IC6. The main function of IC6 is to select either IC5 for the feedback path or an optional multiplexer, the theme of a future article. In the present schematic the multiplexer input is disabled. The output from IC6 goes into IC7 along with the clock pulses from IC8. When the output of the comparator is high so is the output of IC6, then IC7 forces the up/down counters to count down. This continues until the comparator output goes low, at which time IC7 causes the counters to count up. This process is repeated quite rapidly as the clock is operating at about 10MHz. The clock pulses are generated from IC8 with R19, R20 and C11 as passive feedback elements in a TTL loop. These could be replaced by a crystal if a specific operating frequency is desired. The operations of the up/down counters are used to exercise the two 6 bit D/A converters and generate the binary coding. These two D/As, IC12 and IC13, are the heart of the system. Each converts its binary input to an equivalent analog voltage. The analog voltages of each D/A is summed, using different weighting functions by IC14. C12 is used to remove any switching noise that may have reached the converters. R25 is used to adjust the maximum input voltage range, peak to peak. R30 is used for adjusting the A/D for bipolar or biased operations. The output of IC14 is returned to IC5 where it is compared to the input signal and the cycle starts again.

CONSTRUCTION

Component placement is not critical. Make certain the diodes and electrolytic capacitors are properly connected and their polarity is arrested. Use a 40 watt or less soldering iron with fine resin core solder. Apply only enough heat to assure a good connection. All of the parts are standard consumer grade parts. With the exception of R23 and R24 none of the parts is critical and substitutions may be made. All resistors are 20% tolerance and all capacitors are 25 volts, except where noted. For the IC's I would recommend the use of Molex Soldercons as this will eliminate the possibility of heat damage. The power supply can be assembled on the same PC board as the A/D, thus reducing the number of interconnects. The inputs and outputs may be labeled for ease of identification. This can be done with the aid of one of the inexpensive label makers on the market today. The board may be placed inside your computer and use its power supply, hence reducing the need for the two transformers and filters.

SET-UP AND CHECK-OUT

After you have completed the assembly and checked all the regulated voltages, you are ready to

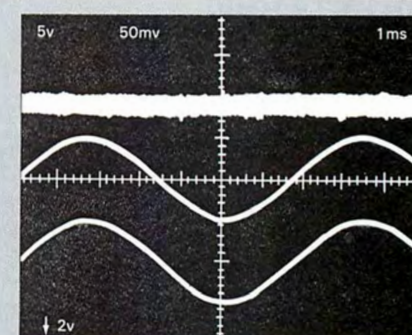
set-up the converter for real time operations. The set-up will be discussed for bipolar operations of the A/D. Other biased operations would be set-up similarly. First ground the analog input to IC4. Then adjust R30 for 5 volt output for bipolar operation, zero volts with respect to circuit ground, for positive operation only or 10 volts for negative input signals. Place a voltmeter on the output of pin #10 on IC14 . The voltmeter should read zero volts, adjust R30 until it reads zero volts, within 1/10 volts or so. Adjust R25 to approximately 1100 ohms total resistance. Recheck the voltmeter and adjust R30 as above. Next, apply a +4 volt DC voltage to the analog input of IC4. Adjust R25 until the voltmeter reads +4 volts. Then ground the input and check for zero volts. If within 20mV proceed otherwise "tweek" R30. Now apply a −4 volt DC level to the input and check the voltmeter. If it doesn't indicate volts try reversing the leads; if it still doesn't indicate −4 volts adjust R25 accordingly. Repeat the zero check again and your A/D should be ready to go.

If you have an oscilloscope connect it to pin #10 of IC14 and apply an approximate 100 Hz sine wave signal with an 8 volt peak to peak level to the analog input. The scope will trace out the reconstructed input signal. Now slowly increase the frequency until the trace breaks, (see Figure 3). This is the maximum frequency sine wave signal that your unit can track for an 8 volt input signal. If you reduce the signal to 1 volt and repeat this you will find the trace now tracks until the frequency has been increased to eight times what is observed for the 8 volt signal. For a 12 MHz clock the A/D can track a 1100 Hz sine wave signal and the settling time to 12 bits is less then 3 microseconds. As adjusted this 12 bit A/D offers resolution steps of 1.22 MV per bit for 5 volt bipolar operation. R25 is used to set the full scale voltage level. If its resistance is set to about 2200 ohms instead of 1100 ohms, as adjusted above, the unit will have a resolution of 0.625 mV per step but will only accept signals that deviate less then 2.5 volts about ground.

This circuit, as can be seen, offers quite a bit of flexibility in its design and operation. Its operation may be optimized to accommodate virtually any type of input signal. In addition if desired its clock and operations may be synchronized with an external clock. After you have interfaced this converter with your computer, by simply connecting the 12 digital bits to the parallel input, you are ready to hook-up a sensor and do some real time processing. Future articles will describe a multiplexer module and various sensor modules to be used with your A/D.

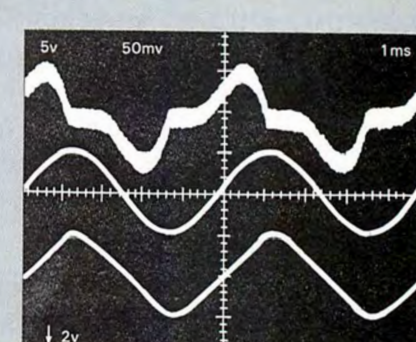
Normal Operation

Comparator Input
Analog Input
Reconstructed Analog Input



Slew Rate Limiting

Comparator Input
Analog Input
Reconstructed Analog Input



Input Over-range

Comparator Input
Analog Input
Reconstructed Analog Input

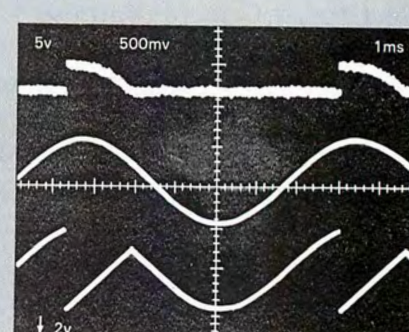
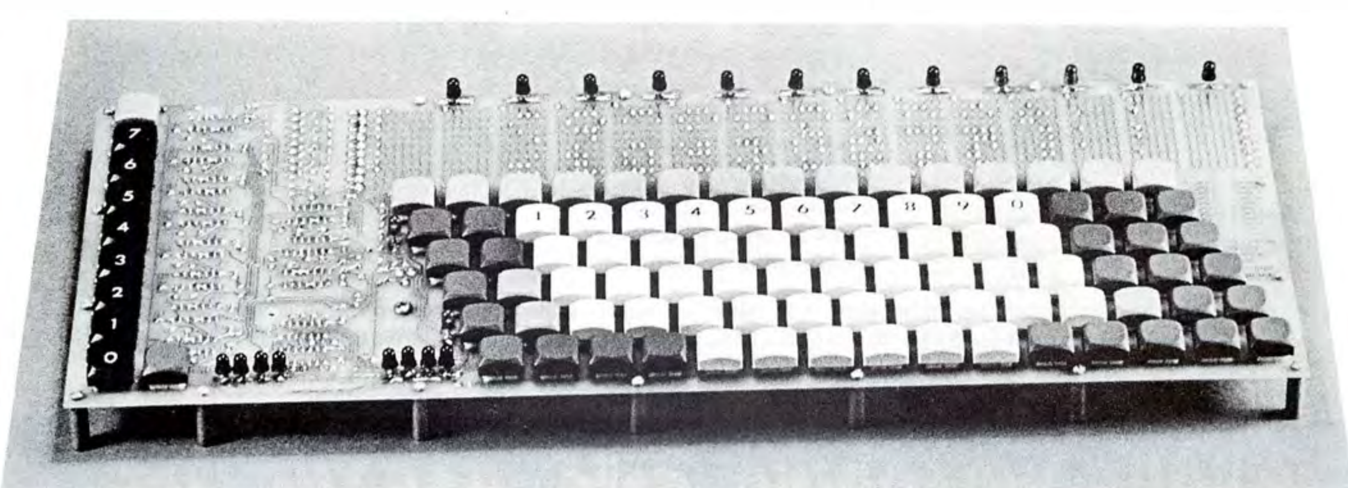


Figure 3.

Versatility
CEI/Custom Keyboards have it



What goes on these keys is up to you

Our aim at CEI is to provide you the hobbiest, with a keyboard system that can perform any task you could possibly demand. Diode encoding coupled with one switch register encoded key allows complete latitude in key definition. Both positive and negative going strobes are provided for interfacing. Visual indication of keyboard status is provided by 22 LED's, 13 of which can have user defined meaning. Shift and shift lock are provided

Unliminted versatility w/100 Keys

90 are encoded off of a diode matrix and are fully user defined.
1 - is encoded by an 8 switch register to allow easy redefination
1 - on board reset key

Diode encoding allows a multitude of encoding techniques. ECIBIC, ASCII, BAUDOT, RADLX 50, FIELDATA or codes of your own design. Complete documentation

Computer Electronics Inc.
37433 Centralmont Place
Fremont, Calif. 94536
Phone (415) 796-5760

☐ Send ____ KB3A Kit. Supplied with all switches, diodes, PC board, TTL-IC's, +5v. regulator, connecting ribbon cable $95.95

☐ Send ____ KB3B Kit. Same as above except without diodes, TTL-IC's, +5v. regulator & ribbon cable. $49.95

☐ Case for above $29.95

NAME ________________
ADDRESS ________________
CITY/STATE/ZIP ________________
NOTE: Calif. residents add 6% sales tax

CIRCLE INQUIRY NO. 4

Resistors — 10% carbon, ½W unless marked

No.	ohms.	No.	ohms.	No.	ohms.
R1	270	R11	220 @ 2W	R21	220
R2	12 @ 5W	R12	3.6K	R22	220
R3	3.6K	R13	3.6K	R23	16.8K 1%
R4	5.4K	R14	3.8K	R24	1.05K 1%
R5	12.3K	R15	2.8K	R25	5K trimpot
R6	27	R16	1.1M	R26	22K
R7	150	R17	22K	R27	51K
R8	3.6K	R18	1.5K	R28	51K
R9	4.6K	R19	20	R29	51K
R10	5.1K	R20	200	R30	10K trimpot

Capacitors:

No.	Specs.
C1, C3, C4	1000 mF @ 25V
C2, C5	.01 mF
C6, C7, C8	.001 mF
*C9	560 pF
*C10, C11	100 pF
C12	1000 pF

T1 secondary 8V to 12V @ 1A
T2 secondary 24V (ct) @ at 1A

CR1, CR2, CR3, CR4, CR5, CR6	1N 4006
CR7	1N 967
Q1, Q2, Q3	2N 3741

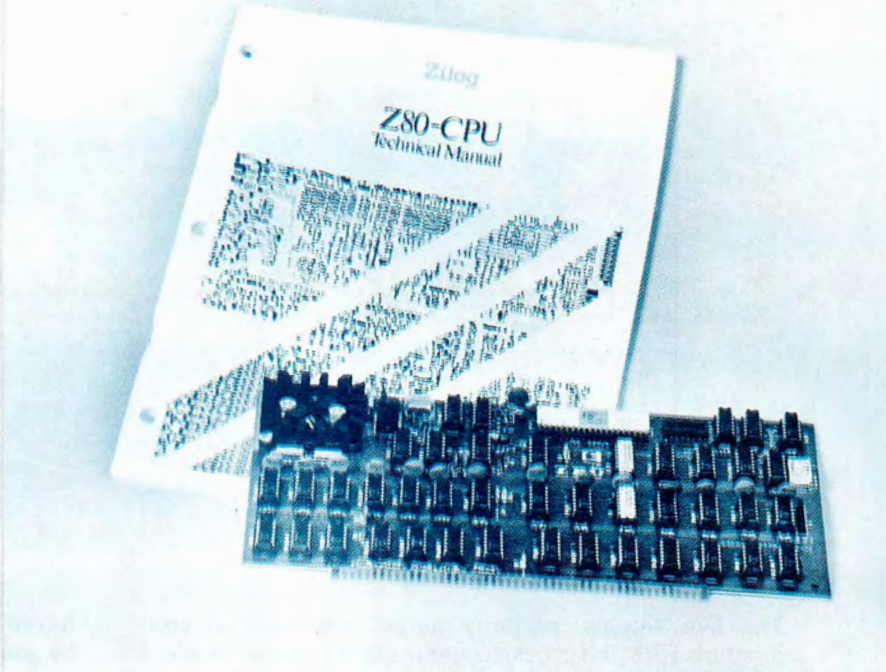
IC1, IC2, IC3	#723
IC4	#741
IC5	#710
IC6, IC7, IC8	#7400
IC9, IC10, IC11	#74193
IC12, IC13	#DAC-01
IC14	#747

Misc.
- P.C. Vector Board 4"x9"
- Line Cord
- On/Off switch
- Chassis box 6"x10"x13½"
- Hook-up wire, solder, etc.
- Parallel interface connector

Note; All of the parts excluding the misc. parts and transformers are available in kit form for $119.50 from Scientific Research Inst.; P.O. Box 83; Marcy, NY 13403. The P.C. Board is also available from this source for $18.00. A P.C. Board plug compatible with the Altair 8800 is available for $21.60.

*These parts may be hand picked for optimum performance.

For most people the Z-80 chip signaled the start of another learning curve in microprocessor technology while others enjoyed the start of a whole new microcomputer revolution



CARD OF THE MONTH THE TDL ZPU™

By Roger Edelson, Hardware Editor

To be fair, this article should have run before my discussion of the other available S-100 bus compatible Z-80 CPU card. The reason is simple: TDL (Technical Design Labs) were out first with their Z-80 CPU board. So this month I will cover the Z-80 CPU card from TDL. I don't know which company, TDL or Sony owns the rights to the name ZPU, nor is my intention to compare the two boards, rather to confine myself to a discussion of the TDL entry. I must say, however, that either board would be a worthwhile addition to your computer.

Let us take a look at the TDL ZPU™ kit before we cover operation and design. The kit is particularly easy to build. My son who is fourteen years of age took about three hours - with time off for ice cream. The board worked the first time out. Assembly instructions are excellent, but please read the paragraph on the identification of the electrolytic capacitor lead polarity. One of the capacitors, C 20, is very strange. It has a single big black dot located in the middle of the two leads. When the dot faces you, the right lead is the positive one. Further on this subject, let me voice a minor gripe: the capacitor is not shown on the schematic diagram. In fact none of the regulator components is shown. This makes troubleshooting this area somewhat difficult. I hope TDL redraws their schematic to include these components.

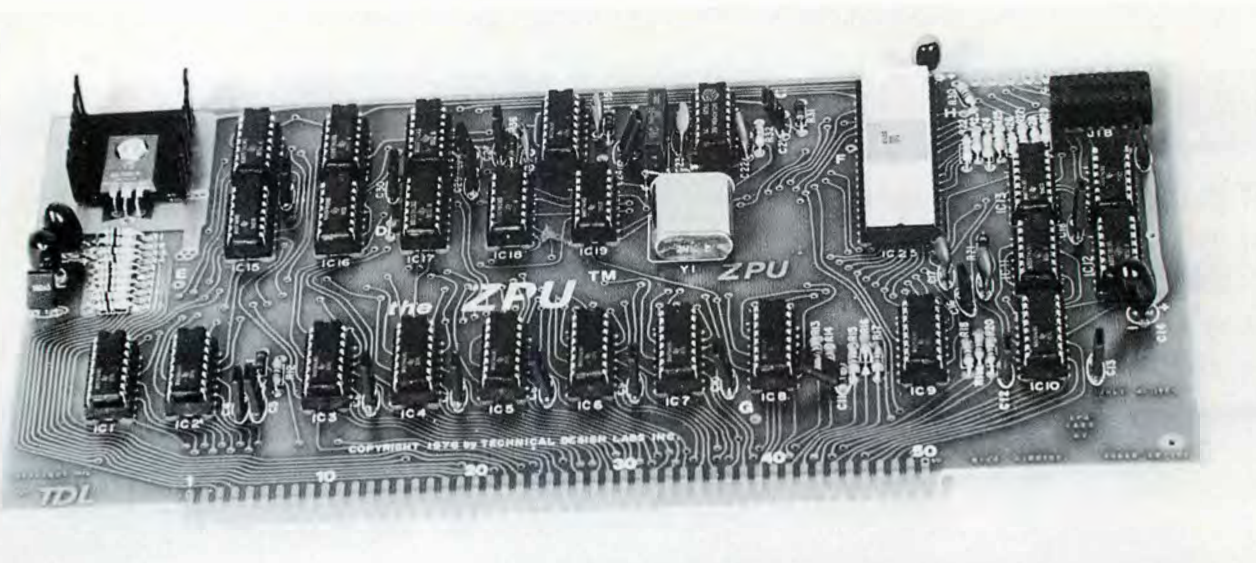
The printed circuit board is high quality with gold-plated edge-connector pins. Sockets are provided for all integrated circuits. A nice touch is the numbering of every tenth pin on the front of the board. The solder masking is generally adequate, but there were some outages on the board front. Where these occurred under a silk-screened component identification, the printing did not take. It is then difficult to read the component identification. With mine most of the resistor numbers on the left side of the board vanished and some identification on the top was also lost. That was a minor problem, but does not detract from the board's construction nor use.

The assembly of the board is straightforward and easy. The assembly instructions contain a particularly good section on the proper procedure for board cleaning. The card is designed to be compatible with the S-100 bus structure or the Altair/IMSAI front panel interconnection. Separate jacks are provided to accommodate the front panel connectors of either computer, and the ZPU user may elect to install either one or both during assembly.

Now let us take a look at the general board design and the specific features provided by TDL. As stated previously, the TDL ZPU™ is S-100 bus compatible. In order to maintain this compatibility the ZPU must generate a number of bus signals not normally produced by the Z-80. The ZPU card produces these signals by logical interaction and gating of the Z-80's status signals and the clock lines.

The most important status generated by the Z-80 are:

1. Memory request
2. I/O Request
3. Read
4. Write
5. M1



The five signals, properly gated, are used in conjunction with the clock to generate all of the required control timing. To follow the description of the logical design of the ZPU, please refer to the schematic of the ZPU card shown on Figure 1.

The Z-80, unlike the 8080, outputs continuous status information whereas the 8080 information is strobed into an 8 bit latch (usually an 8212) during "Sync"time. Consequently, the Z-80 generates no sync pulse. In order to retain the Altair Bus structure, a "psuedo-sync pulse" was created.

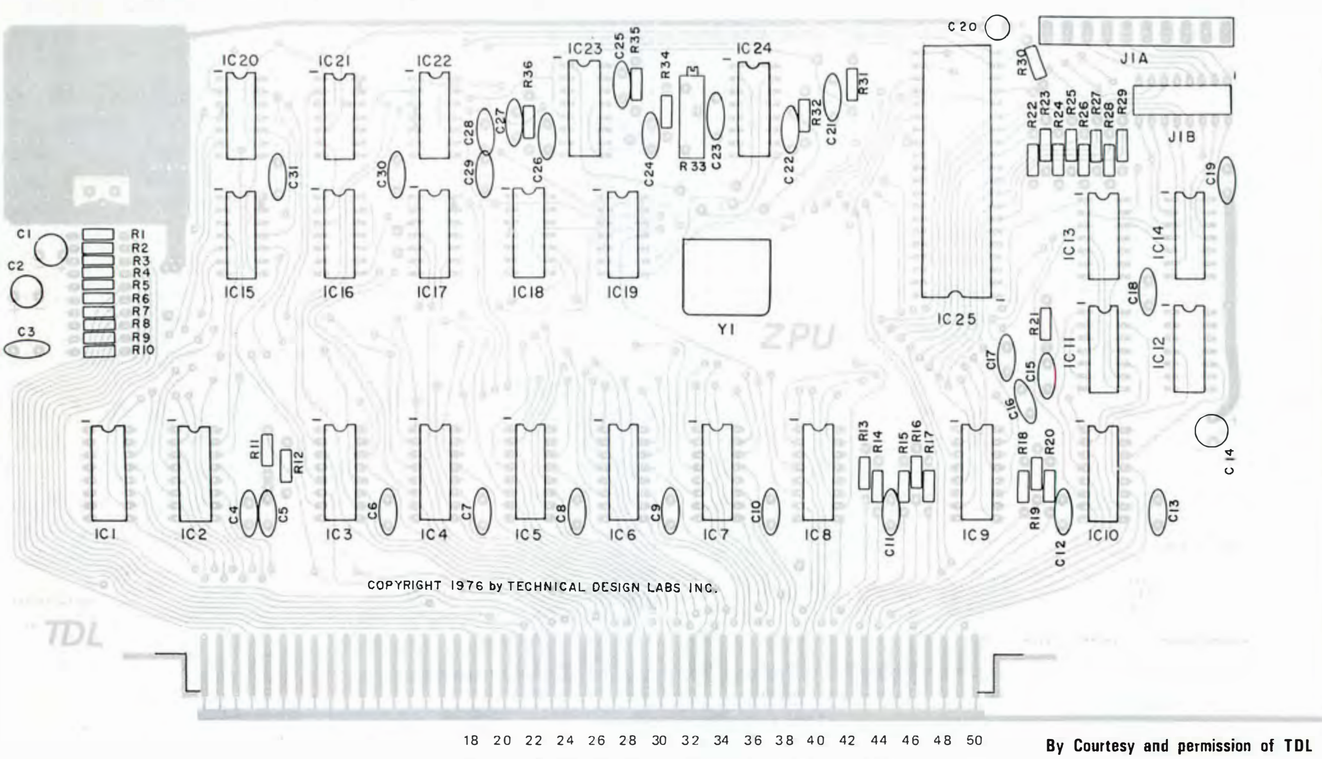
Specifically, PSYNC is generated by gating I/O request and memory request thru a NAND gate (IC21) whose output goes to the input of a 74LS74 (IC16) which is clocked by the Phase 2 signal. PSYNC is taken off of the $\overline{Q}$ of IC16.

A wait is accomplished by gating the PRDY signal and forcing a low into the wait control line of the Z-80. In addition, an extra PRDY line has been made available which may be ***jumpered*** to any unused bus line for future applications. When not in use these lines should be ***jumpered*** together. (pins # 3 and #5 of IC17) The wait signal is initiated by the coincidence of the clock pulse with the pulling down of any of the 3 ready lines (PRDY, XRDY, LRDY).

The interrupts enabled flag is not provided on the Z-80. This has been simulated by the use of an 8 input NAND gate (IC14) and some decode gating (IC17) feeding a set-reset flip flop (IC18) to provide the user with a proper indication when the interrupts are enabled.

The interrupt pin of the Z-80 is handled in exactly the same fashion as that of the 8080, coming to the same bus pin. However, the non-maskable interrupt pin of the Z-80, which represents a significant feature of the Z-80 is brought out to a pull-up resistor, and may be ***jumpered*** to pin #4 on the bus, VI0, the highest priority interrupt line. Thus configuring the Z-80 into the Altair Bus does not detract from this Z-80 feature.

The SSTACK status signal of the 8080 is not gener-



By Courtesy and permission of TDL

IF YOU'RE STILL PLAYING GAMES IT'S BECAUSE YOU HAVEN'T SEEN OUR SOFTWARE LIBRARY

This BASIC SOFTWARE LIBRARY is a complete do it yourself kit. Written in everybodys BASIC immediately executable in ANY computer with at least 4K, no other peripherals needed. Over 1000 pages of source codes, descriptions and instructions.

**** Also available at most computer stores ****

This library is the most comprehensive work done of its kind to date. There are other software books on the market today but they are dedicated to computer games. The intention of this work is to allow the average individual the capability to easily perform useful and productive tasks with a computer. All of the programs contained within this Library have been thoroughly tested and executed on several systems. Included with each program is a description of the program, a list of potential users, instructions for execution and possible limitations that may arise when running it on various systems. Listed in the limitations section is the amount of memory that is required to store and execute the program in K Bytes.

Your Library Is Waiting

The much requested Volumes III & IV are ready. If you already have Volumes I and II you'll want to add these to your set. With the addition of Volume IV there is no reason why anyone who even *THINKS* of using a computer can't own their own *SOFTWARE LIBRARY.* You can start yours for less than $10 plus postage and handling. These programs are *IMMEDIATELY* executable in *ANY* computer with at least 4K. The entire Library is 1000 pages long, chocked Full of Program source code, instructions, conversions, memory requirements, examples and much more. *ALL* are written in compatible BASIC executable in 4K MITS, SPHERE, IMS, SWTPC, PDP, etc. BASIC compilers available for 8080 & 6800 under $10 elsewhere.

**** VOLUME I & II — $24.95 each ****
VOLUME III — $39.95 each
VOLUME IV — $ 9.95 each

add $1.50 per volume for postage and handling. 10% discounts on purchases of any three (3) volumes.
Volume discounts are available to qualified users & dealers.

Personal checks — allow 4 weeks for shipping. Pricing subject to change without notice. If air mail shipping desired add $3 per volume to price, Continental U.S. only. Foreign orders add $8 for each volume.

SRI SCIENTIFIC RESEARCH

1712-I FARMINGTON COURT
CROFTON, MD 21114

Your BANKAMERICARD welcome here

We honor master charge The Interbank Card

PHONES ORDERS CALL
(800) 638-9194

INFORMATION and MARYLAND RESIDENTS
CALL (301) 721-1148

VOLUME ONE

Part 1

BOOKKEEPING
Bond
Building
Compound
Cyclic
Decision 1
Decision 2
Depreciation
Efficient
Flow
Installment
Interest
Investments
Mortgage
Optimize
Order
Pert Tree
Rate
Return1
Return 2
Schedule 1

Part 2

GAMES
Animals Four
Astronaut
Bagel
Bio Cycle
Cannons
Checkers
Craps
Dogfight
Golf
Judy
Line Up
Pony
Roulette
Sky Diver
Tank
Teach Me

PICTURES
A. Newman
J.F.K.
Linus
Ms. Santa
Nixon
Noel Noel
Nude
Peace
Policeman
Santa's Sleigh
Snoopy
Virgin

VOLUME TWO

Part 3

MATH & ENGINEERING
Beam
Conv.
Filter
Fit
Integration 1
Integration 2
Intensity
Lola
Macro
Max. Min.
Navaid
Optical
Planet
PSD
Rand 1
Rand 2

(Volume 2 con't)
Solve
Sphere Trian
Stars
Track
Triangle
Variable
Vector

Part 4

PLOTTING & STAT
Binomial
Chi-Sq.
Coeff
Confidence 1
Confidence 2
Correlations
Curve
Differences
Dual Plot
Exp Distri
Least Squares
Paired
Plot
Plotpts
Polynomial Fit
Regression
Stat 1
Stat 2
T-Distribution
Unpaired
Variance 1
Variance 2
XY

APPENDIX A
BASIC STATEMENT DEF

VOLUME THREE

Part 5

ADVANCED BUSINESS
Billing
Inventory
Payroll
Risk
Schedule 2
Shipping
Stocks
Switch

VOLUME FOUR

GENERAL PURPOSE PROGRAMS
Bingo
Bonds
Bull
Enterprise
Football
Funds 1
Funds 2
Go-Moku
Jack
Life
Loans
Mazes
Poker
Popul
Profits
Qubic
Rates
Retire
Savings
SBA
Tic-Tac-Toe

PARTS LIST

IC	1 TO 10	8T97B
IC	11,12,20	74LS02
IC	13,15,19	74LS04
IC	14	74LS30
IC	16	74LS74
IC	17	74LS10
IC	18,21,22	74LS00
IC	23	74123
IC	24	4049
IC	25	Z80
R	1 TO 10, 12 TO 20, 30	1K
R	11	100 ohm
R	21	47
R	22 TO 29, 32	4.7K
R	33	20 K TRIMPOT
R	34	3.3K
R	35	10K
R	36	12K
C	5	1 Mf
C	15, 17	.001Mf
C	22	10 pf
C	23	6 pf
C	25	27 pf
C	26	47 pf
J1 (A or B)		10 PIN MOLEX or 16 PIN SOCKET

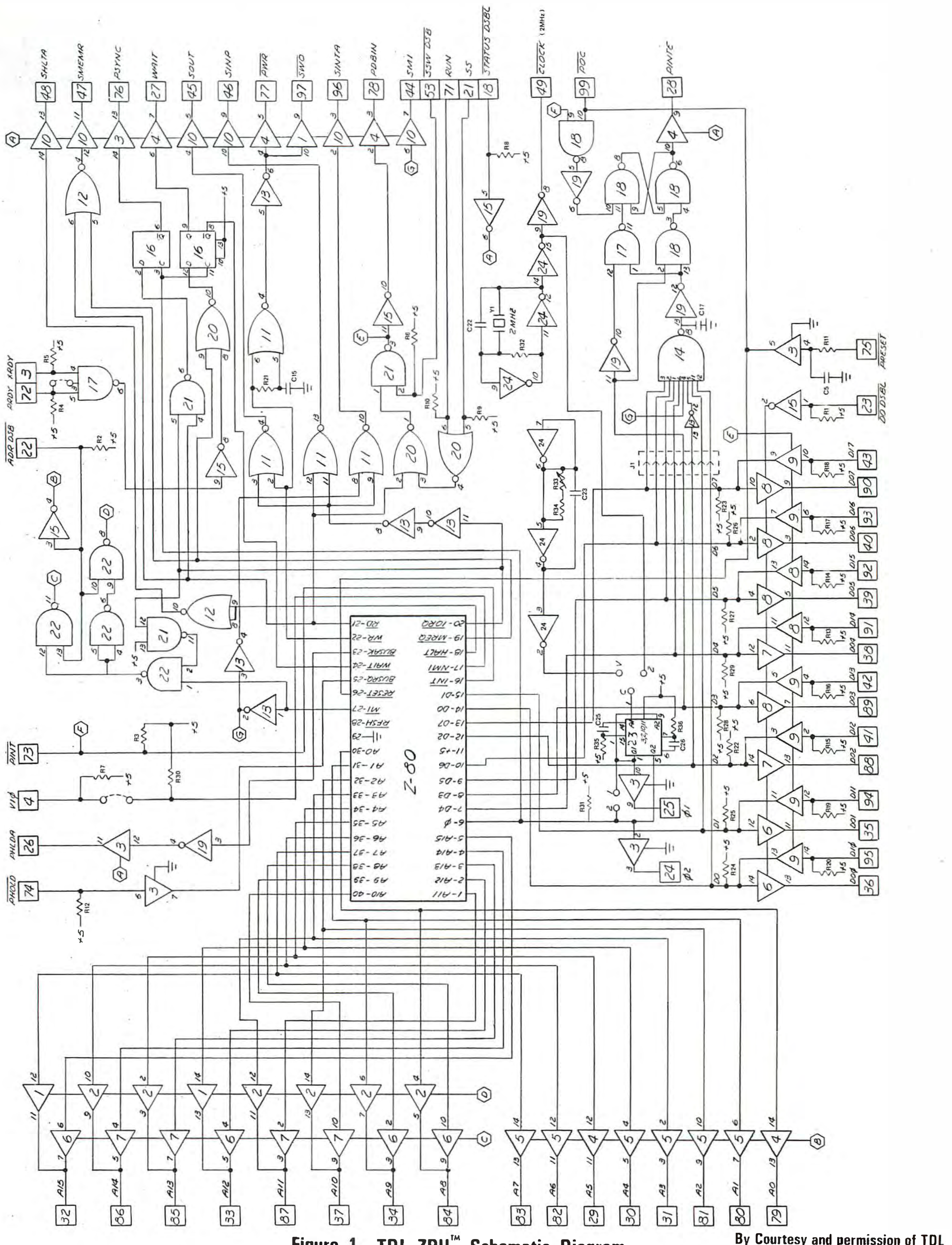


Figure 1. TDL ZPU™ Schematic Diagram

By Courtesy and permission of TDL

ated. Instead, the Z-80 REFRESH signal may be ***jumpered*** out to this line for use with future dynamic memory designs.

Processor write is generated by the Z-80, however in this application TDL has added some additional delay in order that the STATUS OUT or MWRITE may be properly decoded.

Handling of the remaining control timing is straightforward. HALT ACK is generated by the Z-80. The MREAD signal is a function of the Z-80 READ and MREQ signals. STATUS OUTPUT is a function of WRITE in conjunction with an I/O request. STATUS INPUT is a function of a READ in conjunction with an I/O request. PDBIN is a function of the READ signal. The interrupt acknowledge signal is a function of a simultaneous MI and I/O request.

All processor signals with the exceptions of phase one, phase two, and not clock are tri-statable thru the normal Altair bus signal.

The ZPU card features two clocks on-board. The first is fixed at 2MHz thru crystal control, and the second is variable between less than 1 and greater than 4 Mhz by means of a 20 turn trimpot.

The 2Mhz crystal controlled clock is selected by placing a jumper between the augat pins labeled "C" and "2M".

The variable speed clock is selected by "jumpering" between "C" and "V". (The pins "C", "2M" and "V" are located in area A on the ZPU card.)

The crystal oscillator is a parallel resonant circuit using a 2Mhz crystal in conjunction with several gates of IC24, a 4049 CMOS oscillator chip. This clock generates $\overline{\text{CLOCK}}$ and a driving signal for a pair of one-shots. The one-shots (IC23 — no IC number was given on the schematic originally) comprise both halves of a F4123 and are used to generate Ø1 and Ø2 clock signals. This is not a bad way to produce these non-overlapping clock signals.

The variable oscillator utilizes the remaining sections of IC24 in a free-running oscillator whose frequency is controlled by a precision RC network, and the frequency may be varied by adjusting R33, a 20K 20 turn trimpot. The variable oscillator presents phase one and two to the bus. $\overline{\text{CLOCK}}$ is always a function of the crystal oscillator and is always maintained at 2MHz by that clock so that peripheral cards may be made to operate correctly regardless of processor speed. See the section on High-speed operation for details on this.

Regardless of which clock is selected, if the variable clock is tuned to within 100KHz or so of the crystal, there is a tendency for the 2 clocks to "lock in" to each other, that is to get into a fixed resonance. The operational effect of this is that when the variable clock is selected in this condition, initial frequency change either up or down will tend to be resisted until the frequency "jumps" roughly 50KHz, at which point smooth frequency adjustment may be made.

Two augat pins (in area "B" and "C" respectively on the board) are provided for observation of the phase one and phase two signals. These points are test points only and not intended for adjustment of clock speed. Clock speed should always be measured at point C in area A. (On my board area C was hard to find because of the loss of silk-screened information.)

By removing the jumper choosing either of the two on-board clocks and connecting the common pin (C) to an external frequency source, the ZPU card may be synchronized with another system if the user chooses. This also makes it possible to run the processor at very low speeds (down to DC) which on occasion can be tremendously useful. (For example, individual T-states may be observed on the front panel.) This is one of the nice hardware features of the Z-80. It would facilitate matters if a coax connector could be placed at the top of the board to aid in bringing in this external clock.

A visual inspection of the ZPU card reveals more buffers (8T97s or 74367s, ICs 1 - 10) than are usually seen on a CPU card. This additional buffering was necessary to reduce bus loading and to assure normal front panel operation.

The front panels of both the Altair and the IMSAI look at the high order addresses for information about the I/O port number during I/O operations. This was optional with the original designers of the 8080 systems because the I/O port number is output to both the high and low order addresses by the 8080.

The Z-80 outputs I/O port information only to the low order addresses. (Contents of the accumulator are then present on the high order addresses.) Hence, in order for the sense switches to operate normally 8 additional buffers have been added which transfer the lower 8 bits to the high order address lines during I/O operations.

The normal configuration of the ZPU card is that which enables it to operate in an Altair or IMSAI with other peripheral boards.

The kit as supplied and the instructions as given result in a CPU card which may act as a direct replacement for your current 8080 processor. There are however some options which may be exercised by the user which take advantage of several of the Z-80 options. These are:

1. Connecting the REFRESH signal to pin 98 on the bus.
2. Connecting the non-maskable interrupt to vectored interrupt lines.
3. Altering the processor speed.
4. Use of the duplicate PRDY line.

Pin #28 of the Z-80 outputs a $\overline{\text{RFSH}}$ signal, which may be used to provide refresh timing for dynamic memories. This signal may be placed on pin #98 of the bus. Pin #98 is normally occupied by SSTACK on the S-100 bus 8080 system, however this status indicator is not terribly useful and was omitted on the Z-80 altogether.

The $\overline{\text{RFSH}}$ signal may be picked up at area F, immediately to the left of the Z-80, and "jumpered" to the pad in area G, straight down and slightly to the left from the Z-80. This places the signal on the bus.

When the signal is on the bus, the status light on the front panel, labeled STACK will now stay lit when the processor is running, indicating that the REFRESH signal is on the bus.

For the exact timing information about the $\overline{\text{RFSH}}$ signal, see the Z-80 manual. This signal is useful to systems utilizing dynamic RAM storage.

On the Z-80, pin 17 is $\overline{\text{NMI}}$, the non-maskable interrupt. To quote the Z-80 manual:

"The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the

end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80 CPU to restart to location 0066_H."

This powerful interrupt capability is made available to the ZPU user.

Pin #17 of the Z-80 and pin #4 of the bus (VIØ) are normally both held high by pullups. Solder pads at location H and location E may be jumpered together, thus making the NMI available at VIO, the highest priority vectored interrupt line.

The Z-80 has the capability of operating from DC on up to some maximum limit greater than 2.5MHz because of its static nature. To take full advantage of this capability the ZPU card has been designed with a variable speed clock on-board.

P #1, an augat pin soldered to a wire represents the phase one and two inputs to the processor. If the pin is placed in J2, the augat pin labeled "V" in area A, then by adjusting the trimpot located above the crystal, the frequency may be varied over a range of approximately 3 MHz.

Normally, when one is reducing the speeds, simply turning the speed down is sufficient, and no problems will be encountered. For individuals whose systems may currently be marginal at 2MHz, reducing the processor speed may well greatly increase reliability of the system. Some marginal memories may operate with no wait states if the clock is set at about 1.5 MHz.

When speed is increased it is sometimes necessary to readjust the timing of the 74123 for stable operation. This RC network (R36 and C26) effects the phase one and phase two relationships, which become more critical as processor speed is increased.

The duplicate PRDY line was included in order to facilitate operation with the Altair 8800B, or for any other use the user might dream up.

The extra PRDY line comes off of IC17, area D, immediately to the left of the IC has 2 pads which are normally ***jumpered*** together. If one wishes to use the extra PRDY line, remove the jumper, and take the PRDY signal off of pin #3, the top of the two pads.

The 8800B requires 2 additional RDY lines. XRDY2 is on bus line #12. If operation with the 8800B is desired, ***jumper*** the additional RDY line on the ZPU to this bus pin. The other RDY line is FRDY, which is pin #58 on the bus. The user may use this line as he wishes.

The Z-80 unlike the 8080 does not necessarily stop on an M1 state. In order to operate the front panel, the processor must, however, be in the M1 state. TDL has decided to omit additional circuitry which would force the Z-80 to halt at M1. Therefore when operating in single-step mode it is necessary to make sure that you step the processor to an M1 state and then operate the front panel.

The Z-80 chip itself is guaranteed to operate up to 2.5 MHz. Most Z-80's appear to operate in the neighborhood of 3MHz and some will operate at 4MHz. TDL has designed the ZPU board to operate reliably at clock speeds up to 4MHz. The ZPU manual contains an excellent section on high speed operation of the ZPU, including a simple procedure for adjusting the speed of the processor using the TDL SPU to its maximum.

It is important to note that although the system $\overline{CLOCK}$ line is maintained at 2 MHz regardless of processor speed, some boards use Ø1 or Ø2 for their

timing. These boards will not operate correctly when the processor timing is altered. In order to fix this problem, simply cut the clock trace on the non-operating board (from 02 or 02 and *jumper* it to the CLOCK line.) I have done this routinely on all my I/O boards so that I may substitute either the ZPU or the CPU card at will.

The TDL ZPU manual contains a section on compatibility. Bear in mind the fact that there are two types of compatibility, hardware and software. In general, with the exception of the SSTACK signal and the previously mentioned single-step problem, hardware compatibility between the TDL ZPU and the S-100 CPU has been achieved. The front panel of your system will operate in its normal fashion with all switches serving their normal function.

The Z-80 is 100% machine code compatible with the 8080's seventy-eight instructions. Hence standard 8080 software will run without modification on the Z-80. However, if you are using software controlled timing by means of the number of machine cycles necessary to complete a loop, 100% compatibility will not exist. The problem is inherent in the Z-80. The architecture of the Z-80 is more efficient than that of the 8080. In its design many instructions of the 8080 - while having the same machine code, have fewer "T-states": thus the instruction is executed faster in real time.

Obviously where the real time length of a timing loop is controlled by software, the program will have to be rewritten to adjust for the higher realtime execution speed of the Z-80. This is true even if both the 8080 and the Z-80 are running at exactly 2MHz. In the benchmark programs the Z-80 running only the 8080 instruction set has been found to be about 10% faster even while being maintained at 2MHz clock speed.

TDL notes that almost all 8080 languages run without a hitch on the TDL ZPU. The sole exception is Altair Basic. This Basic has as part of its routines several occasions where the parity flag is checked as part of the function. In the Z-80 the parity flag indicates OVERFLOW and not parity during math routines. As a result Altair Basic will not run on the Z-80. However, the structure of the Basic language does not require this use of the parity flag; it was used to reduce program space by several bytes. It therefore can be patched by those who wish to do so. An attractive alternative would be to procure TDL's 8K Basic which is Altair compatible and which contains a large number of exclusive and desirable features. Another solution would be to buy one of the Z-80 Basics now available which provide operational features. They are not, however, 8080 compatible.

The ease of assembly, low cost, and special features of the TDL ZPU make it a good alternative to the standard CPU card. In addition TDL's ZPU manual is very comprehensive and the Zilog Z-80 is also included. TDL also provides a listing and paper tape of their ZAP monitor.

The ZAP Monitor is a IK version of TDL's 2K ZAPPLE Monitor. It is relocatable (can be placed anywhere in memory), expandable ("modules" of additional commands can be tacked on at the end, like cars on a freight train.), and quite powerful as a system executive.

The expandable feature should be of great interest to the user. Since it is designed in a modular fashion, and since the ZAPPLE is its direct parent, this monitor features tremendous expandability - either of routines generated by the user, or by routines provided by Technical Design Labs. Several "modules" which will be of great interest include powerful "breakpoint", "search" and "register display" commands.

For only $269.00 the TDL ZPU represents a very nice low cost means to enhance your S-100 bus system.

(16)

ZPU PARTS LIST

IC 1 to 10	8T97 or 74367
IC 11, 12, 20	74LS02
IC 13, 15, 19	74LS04
IC 14	74LS30
IC 16	74LS74
IC 17	74LS10
IC 18, 21, 22	74LS00
IC 23	74123
IC 24	4049
IC 25	Z-80
IC 26	7805
R 1 to 10	1K, 5%, Brown, Black, Red, Gold
R 12 to 20	1K, 5%, Brown, Black, Red, Gold
R 30	1K, 5%, Brown, Black, Red, Gold
R 11, 31	100 ohm, 5%, Brown, Black, Brown, Gold
R 21	47 ohm, 5%, Yellow, Violet, Black, Gold
R 22 to 29	4.7K, 5%, Yellow, Violet, Red, Gold
R 32	4.7K, 5% Yellow, Violet, Red, Gold
R 34	3.3K, 5%, Orange, Orange, Red, Gold
R 35	10K, 5%, Brown, Black, Orange, Gold
R 36	12K, 5%, Brown, Red, Orange, Gold
R 33	20K, 20 turn trimpot
C 1, 2, 14	47Mf, 25V, dipped tantalum electrolytic
C 3 to 13	.1Mf Disc Ceramic
C 16, 18, 19	.1Mf Disc Ceramic
C 15, 17	.001Mf Disc Ceramic
C20	33Mf, 25V dipped tantalum electrolytic
C 21, 24	.1Mf Disc Ceramic
C 27 to 31	.1Mf Disc Ceramic
C 22	10Pf Disc Ceramic
C 23	6 Pf Disc Ceramic
C 25	27 Pf Disc Ceramic
C 26	47 Pf Disc Ceramic
Y1	2Mhz Crystal
J1A	10 pin molex connector
J1B	16 pin high profile DIP socket
J3, 4, 5, 6	Augat pins
P1	Augat Pin

1 Heatsink
1 ea. 6/32 x 5/16" machine screw, lockwasher, nut
1 ZPU PC board

12	14 pin low profile IC sockets
12	16 pin low profile IC sockets
1	40 pin high profile IC socket

Miscellaneous

6"	jumper wire
5'	solder
1	Zilog ***Z80 CPU Technical Manual***
1	ZPU Documentation Manual
1	Paper tape of the ZAP monitor

CIRCLE INQUIRY NO. 22

COMPUTER COUPLING

By B. D. Lichtenwalner

With a little time and diligence you can turn that old Boudot coded teletype into a reasonable terminal for home computing

Recently several articles reminded us that good old five level, Baudot coded, teletype machines could be used with a hobby computer. With only a few programming steps code conversion could be accomplished. Add a timing loop and the parallel characters from the computer bus could be serialized; but how about the interface to those printer magnets that require about 150 volts to operate effectively? Or how about all those hams that want to link their new computer to their terminal unit and get on the air with a buffered text system, an automated answer back system, or one of those really fancy mail box/replay systems? Here is a circuit that will allow linking teletype loops with open circuit voltages from 100 to 220 volts to your computer without any danger of getting high potential voltages onto the computer chassis. Also the noisy teletype grounds are not tied to the computer chassis. The circuit will provide both "1" and "0" level outputs for mark, and will accept both "1" and "0" level inputs for mark for keying the loop. Provision is also made to have the output blanked for inputs generated by the coupler circuitry.

Circuit Description

Two opto-isolators are used at the heart of the circuit to isolate the loop voltages from the TTL logic compatible interface to the computer. OP1 has the loop current flowing thru it, and as a result turns on the photo-transistor on the device. As a result the output voltage on pin $\boxed{5}$ of the OP1 is low. With the output of OP1 low, the inverter, U1, provides a high level TTL output at circuit output pin $\boxed{4}$. (Boxed Numbers relate to the numbers used on the printed circuit board used in this configuration.) The next section of U1 again inverts the signal and provides the user with a "0" level signal when the loop is conducting. (circuit output pin $\boxed{5}$). When the loop "opens" as a result of keying, or receiving a signal, if tied to a terminal unit TU, the output at $\boxed{4}$ will go low and the output of the circuit on pin $\boxed{5}$ will go high. The delay components (C2 and R3) are used as part of the circuitry to defeat the receiving section from seeing the output from the transmitting section and will be explained later. R2 and the light emitting diode, LED, are used to indicate that the loop is drawing current. It will give the user an indication that the loop is operational, and will blink on and off when the loop keys. It is not necessary and may be omitted if you don't like all that flashing going on while operating.

TO TELETYPE LOOPS

The second iso-coupler, OP2, is used to key the loop (open the circuit). Transistor TL really does the job, and the isolator just provides power to the base to make sure it turns on when the loop should be closed. The circuit works like this: the light emitting diode in the opto-coupler, OP2, is turned on by the current drawn thru R5. This makes the phototransistor in the opto-coupler conduct. Full loop voltage is then applied to the base of the transistor with R9 as a current limiting resistor. When input pin [3] of the circuit is brought to ground the opto-coupler does not conduct, hence removing the forward bias of transistor, T1, and allowing it to turn off. When the transistor is off the loop current drops to the open circuit (Space) condition. It will not quite drop to zero since the circuit does steal a bit of power from the loop to operate. This minimum current is determined by the open loop voltage and the value of R9. The 33K value is great for loop open circuit voltages from 80 to 250 volts. (Range on your machine may drop some on the extreme ends of these voltage ranges.) If you run higher voltages, increase R9. If your loop has an open circuit voltage lower than 80 volts, decrease R9. I have used this circuit to key a 24 volt loop by reducing R9 to about 7.8K. (Use the highest value that produces good results.) If you want to drive the loop with a mark = "1", connect your input to pin [3]. If this is your input pin make sure that input pin [2] is grounded. If you let TTL logic float (open circuited) it will assume the input is in the "on" condition, and the output of U1, pin [4] will always have a "0" level on pin [1] of the opto-coupler. This will always make the loop run "open."

Just one more note about the circuit used in keying the loop. Zener diode D5, is used to prevent excessive voltages from appearing across the collector-emitter junction of OP2. Be sure not to let it out.

The diode bridge consisting of D1-D4 is added to make it convenient when connecting into a loop. I can never remember which polarity is which, so for a few cents I don't have to worry about it. R1 and C1 are transient voltage suppressors to keep out spikes when the loop is going from mark to space. Always a good investment if you are a conservative circuit designer.

Construction

The components used in the project are all available from companies that advertise in this magazine. The total cost of the project for components will be around $10.00. The circuit is easy enough to lay out on perf board, but I prefer to use printed circuits. A circuit board is available from Scott Communications for $3.50. Whether you wire by hand, make you own board, or get the board from Scott the following notes should be helpful in building and getting your unit working. First, make sure the diodes are in the proper direction. If you buy surplus, as I do, it is worth the time to make sure the labeling is correct. A simple ohm meter check is enough. Be especially careful with the Zener. You won't hurt anything if you get it wrong, but your loop will run "open" all the time. Also, assure yourself that the LED is in the right direction; backwards will make the output circuit inoperative.

Testing/Operation

Connect your +5 Volt power supply to circuit pin [1] ground pin [7] to your chassis and power supply. Also for test purposes tie pin [2] to ground temporarily. Connect the loop to pins [8] and [9] and turn on the loop. The LED should glow and the loop should operate as always. If it is running open, check that the input pin to the OP2 (pin [1]) is about +1.5 volts. If it is, check R9 and check that D5 is in the right direction. When the loop is "marking" you can check the keying circuit by opening the ground lead at pin [1] or by shorting circuit pin [3] to ground. Both conditions should make the circuit run "open." When the loop opens, the LED should stop glowing. To check the output circuit, connect a voltmeter from ground to circuit pin [4]. With the loop activated, and the 5 volt power supply connected, there should be a logical "1" on pin [4]. When the loop is keyed (signal received by the TU or pressing a key on the TTY machine) the meter should swing toward zero. Ungrounding pin [2] should give a solid "0" on pin [4]. When circuit operation is verified as correct, move the test lead to pin [5]. With the loop activated, this pin will be at a logical "0," and will go positive when the loop is keyed. Again, ungrounding pin [2] should give you a logical "1" on pin [5].

For some operations it is desirable not to have the output circuit follow the input circuit. To disable the output circuit from signals introduced into the input circuit, connect a general purpose diode between pin [3] and pin [6]. The anode is connected to pin [3].

Conclusion

The circuit shown provides a simple but effective way to isolate high voltage TTY loops from normal TTL

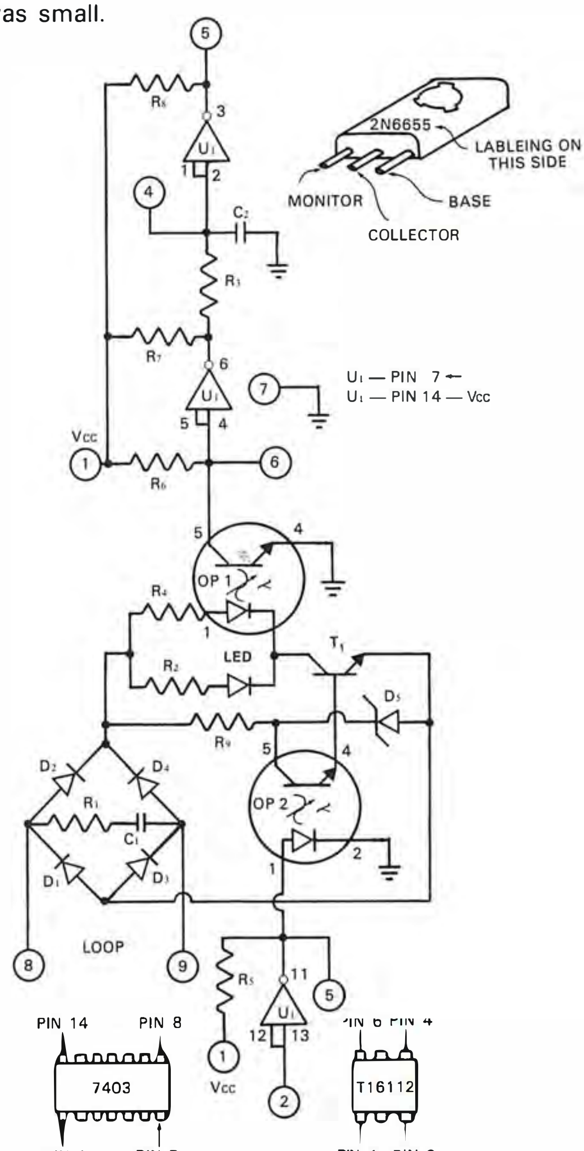
level circuits. It has the flexibility of being keyed by both polarity signals and also provides TTL level outputs with both a positive and negative signal for marking conditions. Commonly available parts and available circuit board makes this an inexpensive and easy-to-duplicate project.

Design Note

Values for R9 are calculated as follows

R9 = V OPEN LOOP −3+[R LOOP X 0.06] 0.003

The original design was done for a loop with 100 volts open circuit and an assumption that the R LOOP was small.



PARTS LIST

U1 — 7403

D1, D2, D3, D4 — IN 4005, 600 PIV, 1 AMP DIODE

D5 — IN 4732 — 4.7V, 1 WATT ZENER DIODE

OP1, OP2 OPTO — COUPLER, TI TIL 112

LED — SMALL RED LED SIMILIAR TO MVD 50

C1 — 0.1μf, 600V CAP

C2 — 0.01μf, 10V CAP

R1, R5 — 470ohms, ¼ WATT R.S.

R2, R3 — 150ohms, ¼ WATT R.S.

R4 — 47ohms, ¼ WATT

R6, R7, R8 4.7K, ¼ WATT

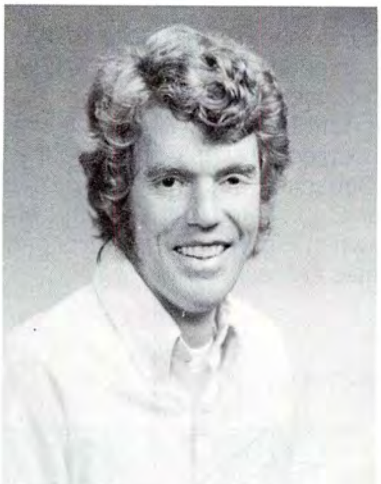
R9 33K, ½ WATT

T1 = 2N5655

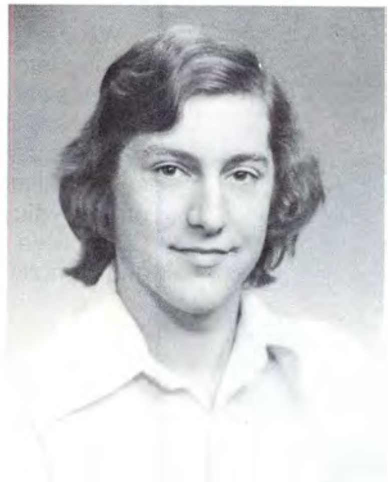
◯ INDICATES TERMINAL CONNECTIONS FOR CIRCUIT INPUT/OUTPUT

FIGURE 1

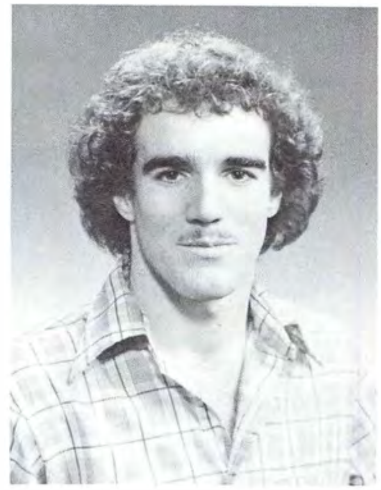
THE 1976 COMPUTER STORE ALL-STARS



Mike Eusey

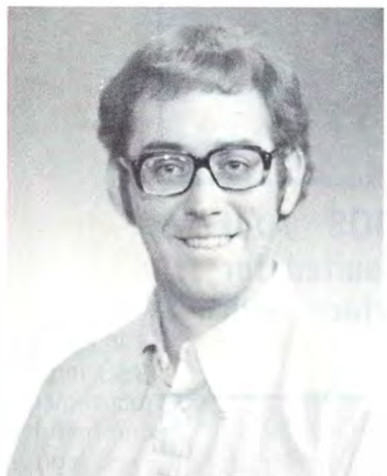


Steve Zook



Paul Tripe

John Trotter



Jim Modecki

Dick Heiser

The most experienced computer dealers anywhere

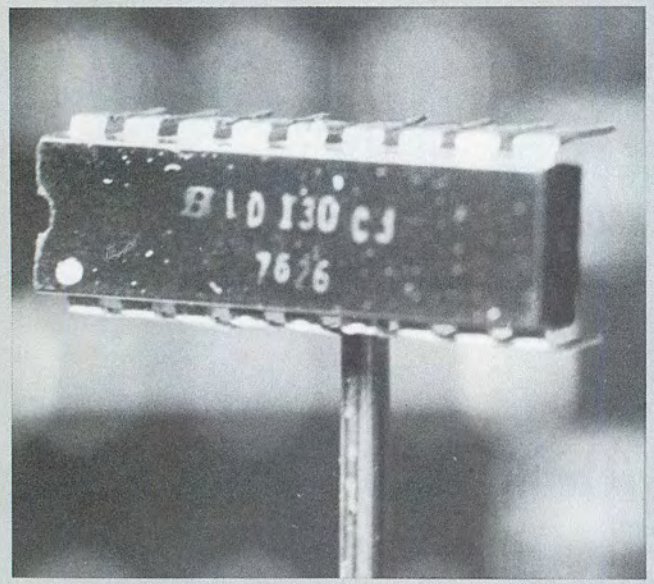
THE COMPUTER STORE

820 Broadway
Santa Monica, Calif. 90401
Phone (213) 451-0713

mits altair™ COMPUTER CENTER

Authorized dealer for:
MITS, Southwest Technical, Polymorphic, Cromemco, Microterm, Oliver, TDL, Sanyo
BY POPULAR DEMAND! NEW HOURS!! TUES – FRI Noon - 8 p.m., SAT 10 a.m. - 6 p.m.
Located 2 blocks North of the Santa Monica Freeway at the Lincoln Blvd. exit

APPLYING



The Siliconix LD 130 CMOS A/D Converter chip is admirably suited for providing a general A/D interface with your microprocessor.

We shall open this month's discussion of the LD130 by looking into the selection criteria for the external components and reference voltage.

Any sampling rate from 1 to 60 samples per second can be accommodated by simply changing the Integrator and Oscillator Capacitors (C_{INT} and C_{OSC}). To find the proper value for C_{OSC}, refer to the clock frequency vs C_{OSC} curve shown in Figure 1. The oscillator frequency and sampling rates are related by:

$$\text{Sampling Rate} = \frac{f_{OSC}}{6144}$$

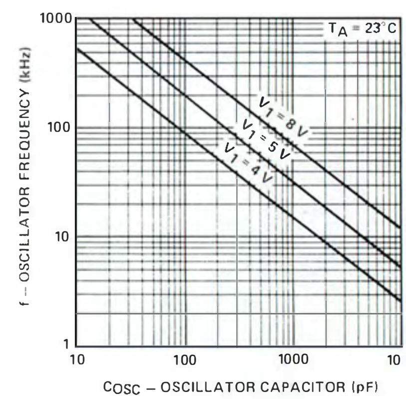


Figure 1. f_{OSC} vs C_{OSC}

In some applications it may be desirable to drive the OSC input with an external clock. This is a common requirement in many data acquisition systems. The LD130 can be driven by an external clock capable of going to logic high and low levels of $V_1 - 0.25$ V and 0.25 V respectively while sourcing or sinking 1 mA. This drive requirement is needed to override the drive capabilities of the CMOS oscillator input shown in Figure 2. A standard TTL gate will not be capable of providing this time requirement.

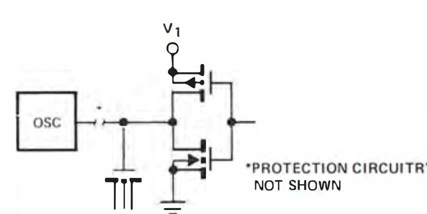


Figure 2. Oscillator Input

The integrator capacitor must change as a function of frequency by means of the following relationship:

$$C_{INT} \cong \frac{10^3}{f_{OSC}} \mu F/sec$$

THE LD130

By Roger Edelson, Hardware Editor

The "Quantized Feedback" conversion scheme introduced by Siliconix provides the LD 130 with AUTO-ZERO Auto-Polarity from a single reference voltage.

Part II of a Series

Minimum supply voltages for operation are $V_1 = 4.3$ V, $V_2 = -4.3$ V. Although the LD130 will be functional at these voltages, TTL compatibility can no longer be guaranteed. Maximum voltages for functionality are $V_1 = 8$ V, $V_2 = -8$ V. V_1 and V_2 must be matched within 1.5 volts of each other or a zero offset will be created.

One of the major features offered by the CMOS LD130 is its very small power consumption (25 mW @ ± 5 V). This power consumption, however, increases significantly as V_1 and V_2 approach their maximum rated limits (60 mW @ ± 8 V). Low-power operation then dictates a lower supply voltage.

There exists a number of power supply and board layout problems that can lead to instability and poor performance in the LD130 A/D system. The problems arise from two main considerations—V_1 regulation and ground system layout.

V_1 regulation becomes important to the LD130 for a number of reasons. First, the LD130 linear stages are powered by the same positive voltage that supplies the logic section and, if used, a LED display. Display noise on this supply then must be rejected by the amplifiers. Since the LD130 has excellent power supply rejection for V_1 (0.6 mV change of output for a 1V change in V_1) this error source will be considered to be of secondary importance.

A major consideration is the modulation of the internal oscillator frequency by an unregulated V_1. A glance at the C_{OSC} vs f_{OSC} curves of Figure 1 shows that this oscillator frequency is a strong function of V_1. The V_1 seen by the LD130, however, can carry a considerable A.C. component when LED displays are used.

When interfacing with a microcomputer, the display problem does not exist. Proper attention should still be paid to grounding techniques to ensure that no logic currents are imposed on the LD130 ground return line. Furthermore the LD130 +5V supply should be well bypassed to ground and either a series diode or a regulator should be used to provide some series resistance on the capacitor filter. (See Figure 5)

The Auto-Zero capacitor (C_{AZ}) is not critical and should remain at 0.1Mfd over the full sampling rate range. In special cases of combinations of low sampling rates and high ambient temperatures (not usually found in home environments) a higher value may be necessary. Capacitor tolerance or type is not critical; standard ceramic disc types may be used.

Capacitors C_{AZ} and C_{INT} should be of a type which maintains a high insulation resistance over the temperature. All Film Type capacitors are suitable.

The equation $\text{COUNT} = \dfrac{\text{VIN}}{\text{V}_{\text{REF}}}$ given last month shows that a 2.000V reference is required to

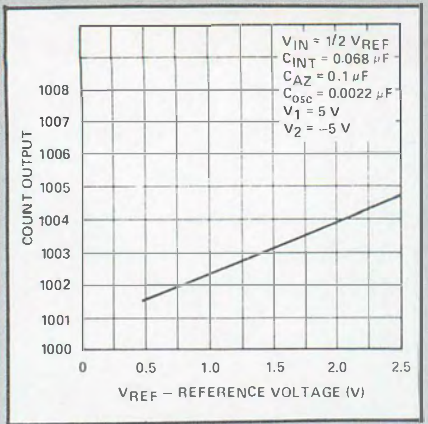


Figure 3. Ratio Operation

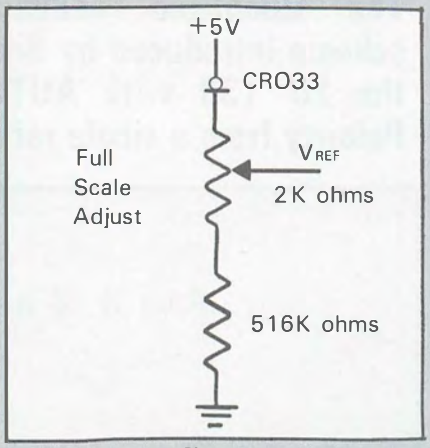


Figure 4.

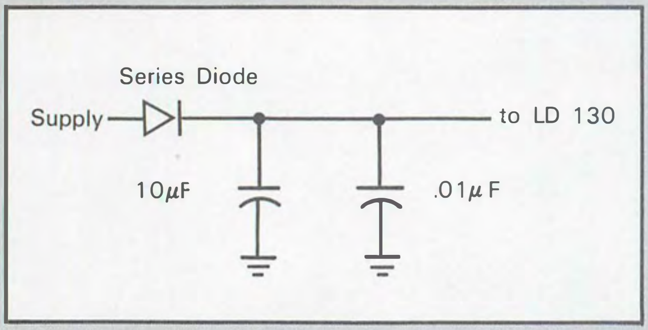


Figure 5.

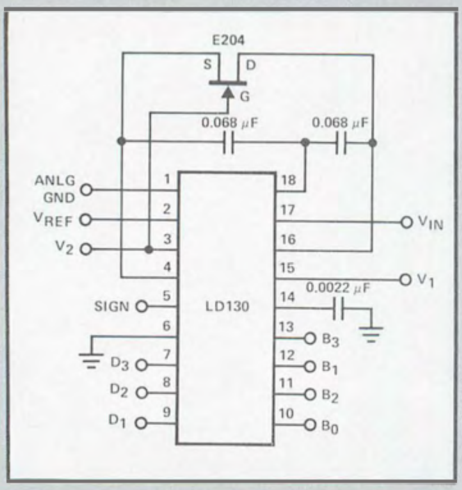


Figure 6. Lock-Up Protection FET

achieve an output that reads directly in millivolts of analog input (1.000 Volts Full-Scale). Scaling the output to read in other than volts can be achieved by adjusting V_{REF} according to this equation. It must be realized that the actual LD130 can vary by as much as 5% of the predicted values, so the V_{REF} must be adjustable for proper calibration.

Since the output of the LD130 is simply the ratio of the input voltage with the reference voltage, ratio measurements are natural to this system. The typical ratio operation error of the LD130 is depicted in Figure 3. A lower V_{REF} limit of 0.5V is recommended for a stable jitter-free output. Reference voltages less than 0.5 Volt provide an unstable reading due to the large amount of noise inherent in the CMOS op-amps.

The typical LD130 application as a 3 digit DVM shows the reference voltage being developed by a current regulator diode-resistor combination. (see Figure 4.) This is the preferred method for creating stable low-voltage references. The temperature compensated current regulator diode (the electrical dual of the temperature compensated Zener) keeps a constant current through the series connection of metal film resistor and cermet trimmer. A typical temperature coefficient of 50 ppM/°C is achieved by this system—an order of magnitude improvement over a typical low-voltage Zener (IN746, 3.3 V Zener).

The LD130 specifications limit the maximum input current at V_{REF} or VIN to 1ma. In order to prevent damage to the chip a current limiting resistor should be placed in series with these inputs if the voltage might exceed the supply voltage by more than 0.3V. A 250 K^{r} resistor in series with pin #17 will offer input protection up to 250V over-voltage and accuracy would be degraded by only 0.025%.

The LD130 has protection circuitry at all inputs and output which prevent static damage by clamping the voltage at these pins.

The E204 JFET shown in the LD130 application of Figure 6 eliminates a power-on lock-up mode. This condition manifests itself by a constant 007 output. If the JFET is not used it may be necessary to recycle the power supplies to attain normal operation. The use of the E204 JFET reduces the maximum allowable negative supply voltage, V_2 to -5.5 V. A higher V_2 supply may be used if a higher pinch-off FET is used (E211 extends V_2 to -6.5V).

While both the Digit Strobe (D_1, D_2, and D_3) and Data Bit (B_0, B_1, B_2, and B_3) outputs are capable of driving 1 TTL load, the maximum internal clock stability and thus A/D stability is attained when the bit outputs sink less than 400 A. Therefore, CMOS or low-power Schottky TTL decoders are preferred. Standard TTL loads will not contribute to A/D instability when an external oscillator is used.

The interdigit blanking period allows the LD130 to interface with gas discharge displays when oscillator frequencies of 16 kHz or less are used.

Since the BCD data for each digit appears before and does not change until after the digit strobe (except when new data are loaded), interface problems such as latching of improper codes are minimized.

The net digit duty cycle is reduced to 25% by the interdigit blanking period. Average LED currents must be calculated with this consideration.

$I_{AVG}\,(LED) = Zi_{PEAK} \times 0.25$

Since the total inter-digit blanking time is equal to a digit ON time, it may be used as a fourth digit strobe for special applications.

The ranging signals (overrange and underrange) are time multiplexed on the SIGN/UR/OR output. Demultiplexing consists of logically ANDING this output with either the D_1 or D_2 Digit Strobe output (see Figure 7). Thus:

D_1 • SIGN/UR/OR = Underrange Pulse (active high) when count < 80

D_2 • SIGN/UR/OR = Overrange Pulse (active high) when count > 999

D_3 • SIGN/UR/OR = Plus polarity sign

If either an underrange or overrange condition exists, the appropriate pulse will occur once each sampling interval during the zeroing time. This single pulse can be used directly to step an Autoranging circuit into the next range. Figure 8 shows the implementation of an Autoranging System for the LD130.

The LD130 offers features useful in many data acquisition system applications. The compact size, low parts count, low power and BCD output of the LD130 can become important features in the trend to complex portable instrumentation. The integrating type "Quantized Feedback" conversion technique gives high noise rejection properties while offering conversion rates of up to 60 per second.

The fact that the LD130 offers mixed Measurement and Zero periods suggests that high rejection to specific noise frequencies can be obtained. While this is true, the rather variable nature of the LD130's internal oscillator makes it impractical to reject specific problem frequencies (such as the A.C. line frequency). An external oscillator can allow this maximum rejection (70 dB) to be attained by providing a more precise clock frequency. The proper oscillator frequency can be chosen from the following relationship:

$$f_{OSC} = \frac{2048\, f_{noise}}{N}$$

where N is any integer that keeps the oscillator frequency within its prescribed limits (6kHz to 370 kHz).

The worst case Normal Mode Rejection for an untuned system is defined by:

$$\text{NMRR (worst case)} = -20 \log \frac{f_{OSC}}{4096\, \pi\, f_{noise}}$$

The internal latches of the LD130 can be made to hold a reading indefinitely when a SPDT switch is added to the circuit as shown in Figure 9. In the convert position, the Auto-Zero Capacitor, C_{AZ}, is connected to the AZ input and the internal latches are updated after each conversion. When switched to "Hold", the large negative output of the AZ amplifier drives the Integrator to the positive rail. Since the comparator can never go low, the control logic retains the data from the conversion preceding the switch change to "Hold". While the "Hold" mode, the Auto-Zero function is inhibited. This means that V_{AZ} is no longer valid and that the first conversion after returning to the "convert" position will not be valid.

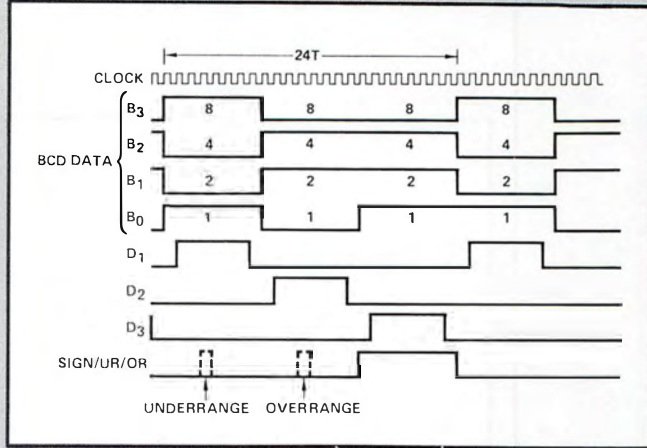


Figure 7. Data Output Format (Output = 769)

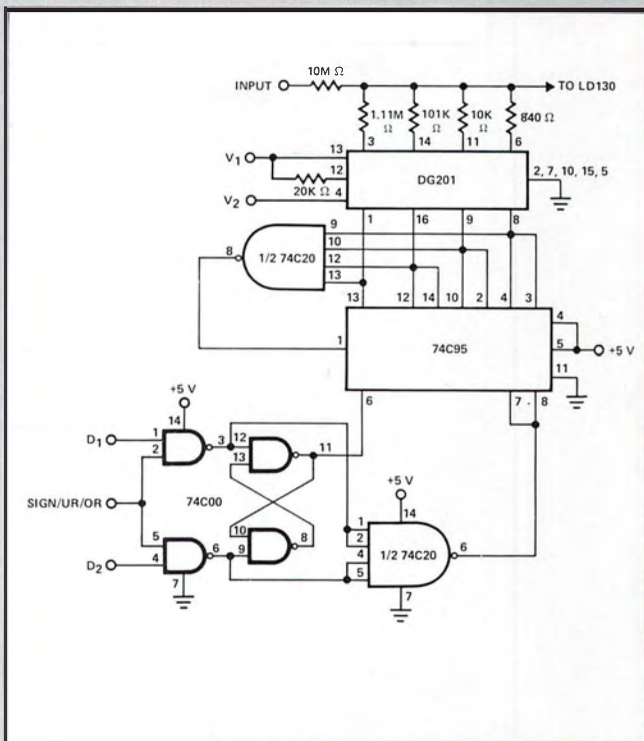


Figure 8. Autoranging System For LD130

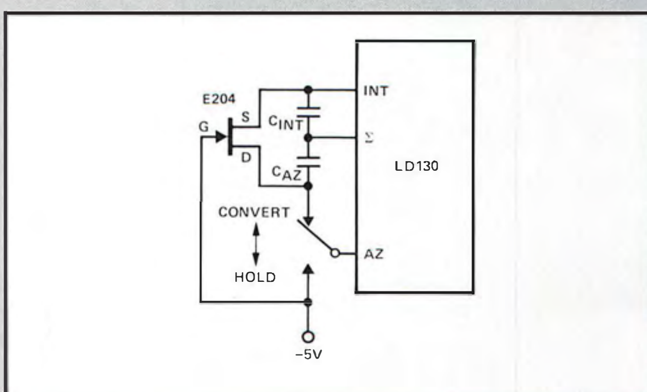


Figure 9. "Hold" Function

Note that the AZ line (pin #16) must be switched to −5V. Standard TTL logic can not be used in this case, but a SPDT analog switch operating from ± 5V supplies could be used. The Siliconix DG 300 series should provide this capability while allowing interfacing with standard TTL logic at its control pin.

Due to the limited number of pins available on the LD130, the internal system timing signals (M/Z, AZ) are not available to the outside. In the instances where A/D system timing may be needed, it may be decoded with the circuit of Figure 10. This circuit uses the integrator output (INT, Pin 4) to develop an End-Of-Conversion (EOC) signal that approximates the complement of the internal AZ signal depicted last month. The circuit can decode these states because the integrator operates around 0 volts during the Measurement and around −1 Volt during the Auto-Zero phase. The average value of the integrator then is indicative of the state of the AZ signal. The LD130 data are valid and remains valid as long as the output of the EOC circuit is "0". Switching of an analog multiplexer should occur during this time.

It is important, for the proper functioning of this circuit over the full analog voltage range, that C_{INT} be increased to twice the value given by Equation 2.

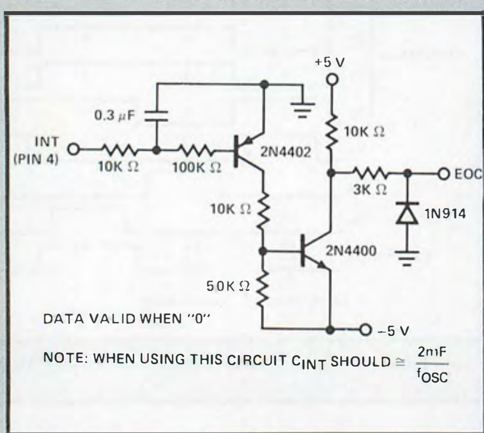


Figure 10. End-of-Conversion Decode

Although multiplexed BCD is more generally useful, the output of the LD130 can be demultiplexed and presented in a parallel format as shown in Figure 11.

Applications for this format include printer drive and interface to a 12 bit data bus. Interface to a printer would also require the presence of the End-Of-Conversion Signal as a Print enable. The EOC decode circuit is included in the schematic for this purpose. Multiplexed-to-Parallel BCD conversion is accomplished by the three CD4042 Quad latches. These CMOS ICs minimize parts count by eliminating the need for data bit buffers.

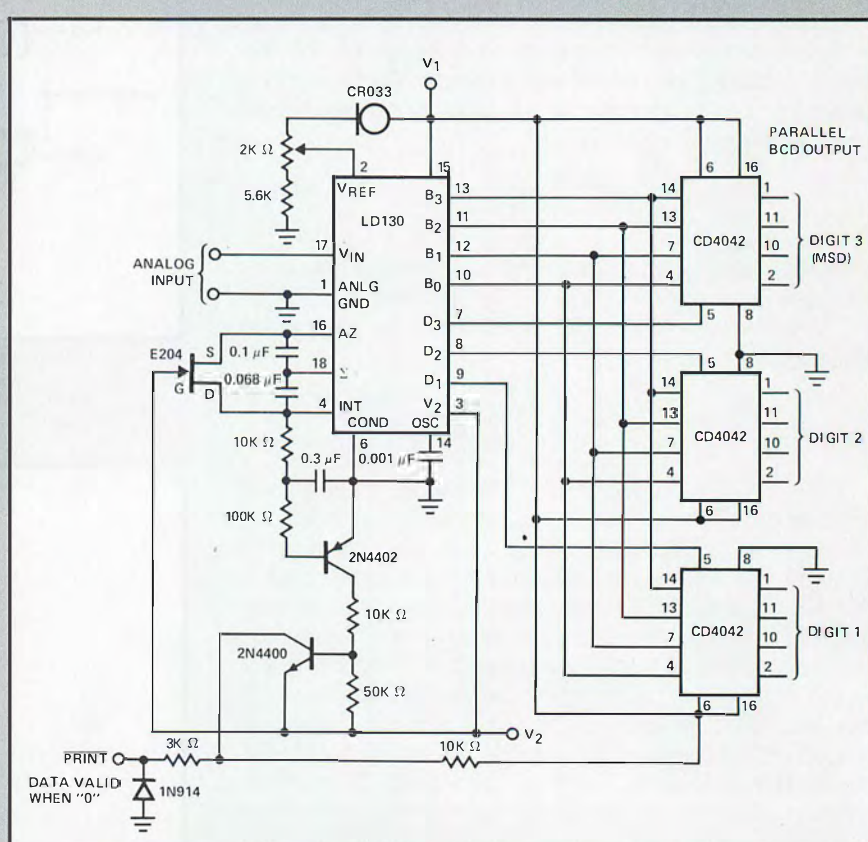


Figure 11. 3 Digit A/D with Perellel Output

The A/D- μP system is the preferred method of handling and storing analog quantities. The cumulative errors of analog data handling are eliminated while powerful system functions can be achieved with a few ICs.

The LD130 A/D has many features important to μP-based systems such as:

1. *BCD Coding* — The must useful format when the output is to be observed by humans.

2. *LSI* — Keeps system complexity down.

3. *Multiplexed Output* — Readily interfaces 4 and 8 bit systems.

One of the problems involved in interfacing the 4 bit μP is that the 4 bit bus is fully taken up by the 4 BCD data bits. Additional bits for digit markers are not available. One solution would be to apply the digit strobes to an additional input port thus identifying the data. This method, shown in Figure 12, offers a simple hardware interface for the LD130 but requires an additional input port and a significant amount of processor time to synch up to the non-synchronized LD130. Here the circuit shown in Figure 10 is used to provide an end-of-conversion (EOC) signal to the μ P. The μ P must constantly monitor the input port containing this signal and use the information presented from D_3, D_2 and D_1 to determine which digit is present.

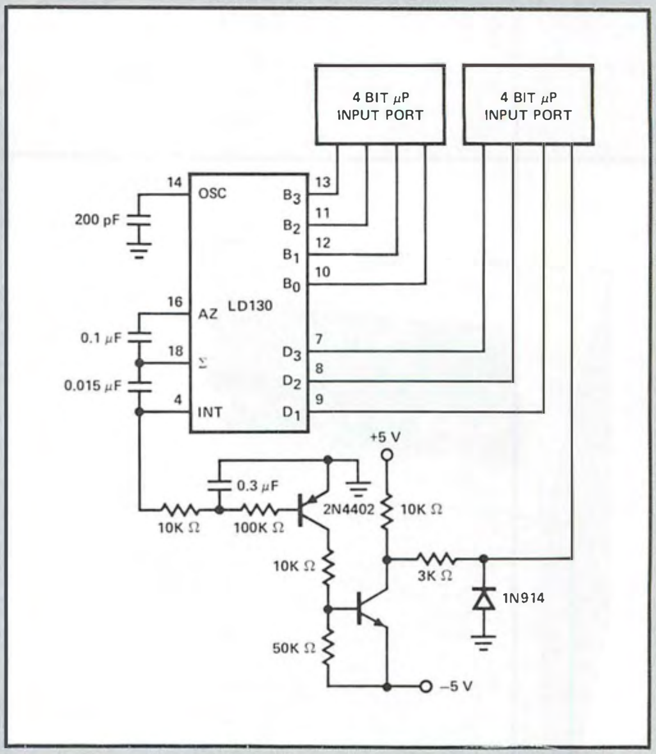
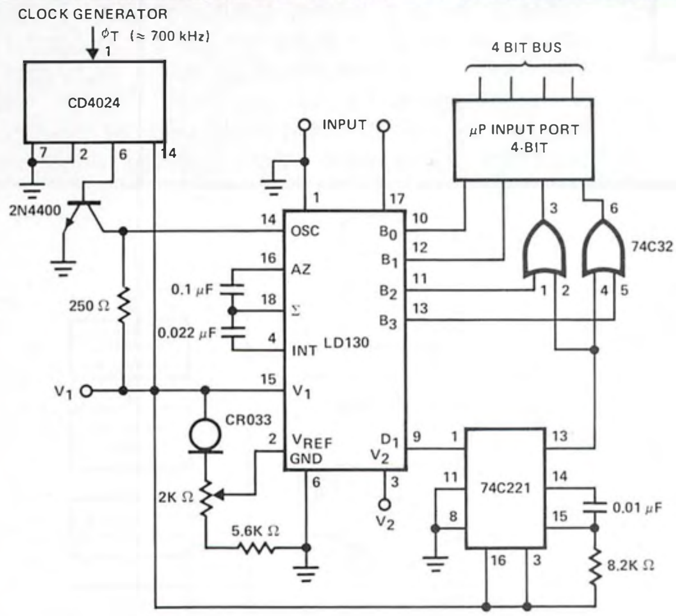


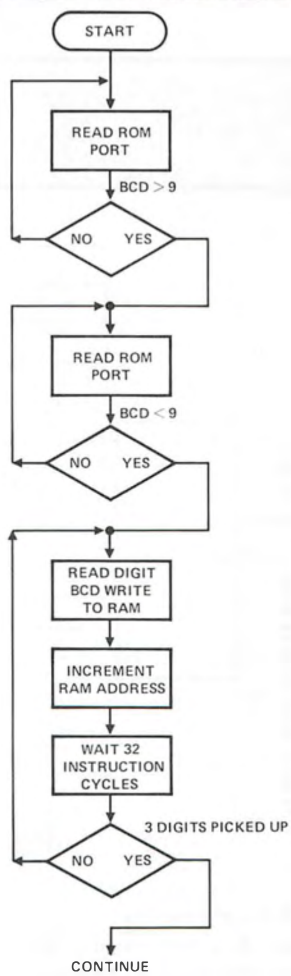
Figure 12. 4 Bit μP Interface



a) Synchronized 4 Bit μP Interface

```
          FIM ϕ, ϕϕH     ; RAM AND ROM ADDRESS
          FIM 2, 6ϕH     ; ADDER CONSTANT
          FIM 4, DϕH     ; TIME CONSTANTS
          SRC ϕ          ; REGISTER CONTROL
          CLB            ; CLEAR CARRY AND ACCUMULATOR
READ:     RDR            ; READ ROM PORT ϕ
          ADD 2          ; ADD CONSTANT (6)
          JNC READ       ; REPEAT IF D1 HASN'T OCCURRED
CHANGE:   CLC            ; CLEAR CARRY
          RDR            ; READ ROM PORT ϕ
          ADD 2          ; ADD CONSTANT (6)
          JC CHANGE      ; LOOK FOR FALLING EDGE
NEXT:     RDR            ; READ 4 BCD BITS OF D1
          WRM            ; WRITE TO RAM
          INC 1          ; INCREMENT RAM ADDRESS
          SRC ϕ          ; SEND NEW RAM ADDRESS
TIME:     ISZ 5, TIME    ; TIME DELAY
          ISZ 4, NEXT    ; READ NEXT DIGIT
```

b) Program Instructions

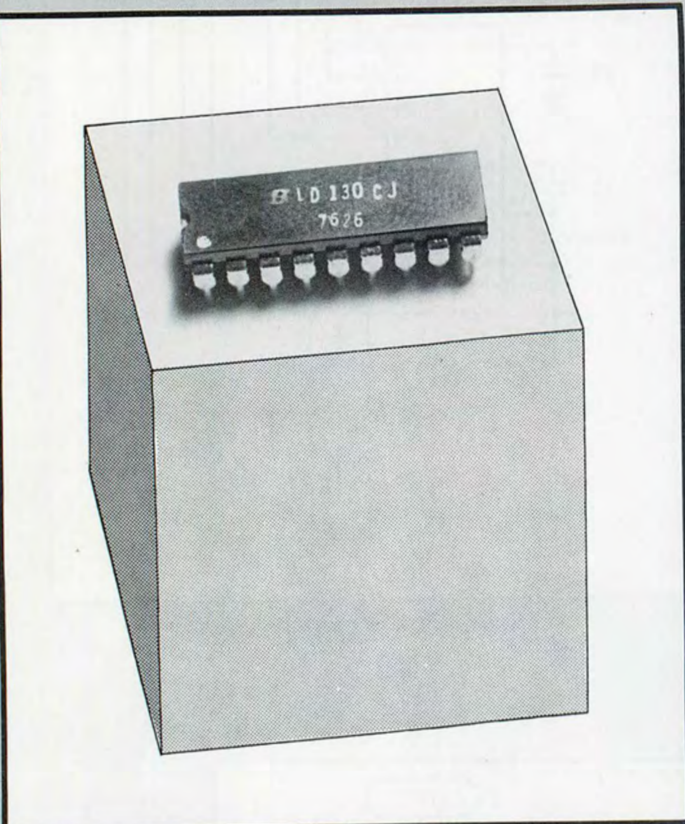


c) Flow Chart

Figure 13. 4 Bit μP

The interface shown in Figure 13 simplifies software and saves μP polling time by synchronizing the LD130 to the μP system clock. This application is designed specifically for the INTEL 4004, 4040 μPs. As shown in Figure 13a, the μP system clock is divided by 16 (by the CO4024 counter) before driving the OSC input of the LD130. The D_1 output of the LD130 triggers the one-shot which, in turn, forces an unallowed BCD state (11xx or $\geqslant C_h$) on the data bit lines—this is the cue that allows the μP to identify D_1 time. When the unallowed BCD state disappears, the μP picks up the D_1 data and stores it in memory. D_2 and D_3 data are picked up 32 and 64 μP machine cycles later respectively. Since the μP and the LD130 are synchronized by the common clock generator. The flow chart and software for this system are given in sections b and c of Figure 14.

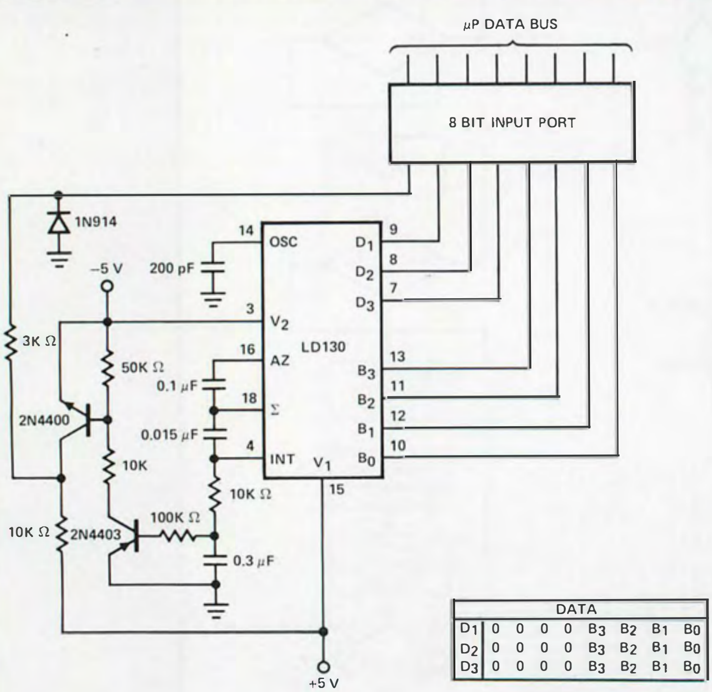
Other counter types—74163 for example may also be used as may other one-shots (74123).



Obviously, the statement "3 Digits Picked Up" will have to be implemented by either incrementing or decrementing a register and checking the result. Maximum line-frequency rejection can be an added bonus of this system if the μP clock oscillator frequency is chosen to reject this specific noise frequency.

The interface shown in Figure 13 does not include the data valid (EOC) signal. This could be added to the system and the output logically "ANDED" with the output of the one-shot. An alternative approach would be to simply repeat the pickup routine a second time to verify the data.

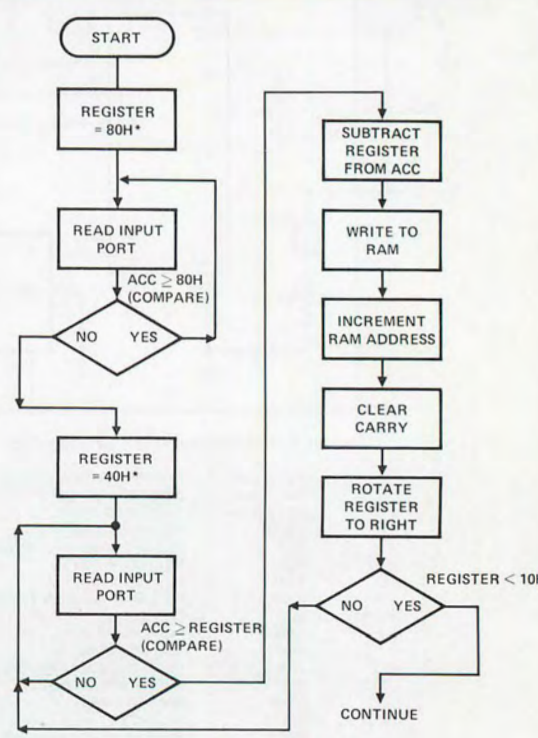
The 8 bit bus allows additional data markers to accompany the BCD data to the μP accumulator. Figure 14a shows all 8 lines taken up with 4 BCD data bits, 3 digit strobes and a data valid bit. This data interface requires the μP to occasionally poll the input port to determine if the data valid bit is low (input is less



a) 8 Bit μP Interface

	DATA							
D1	0	0	0	0	B3	B2	B1	B0
D2	0	0	0	0	B3	B2	B1	B0
D3	0	0	0	0	B3	B2	B1	B0

b) Data Storage Format



*H DENOTES HEXADECIMAL

c) Flow Chart

Figure 14. 8 Bit μP

than 80H). When this occurs, the μP then successively looks for the appearance of D_1, D_2, and D_3 by comparing with the numbers 40H, 20H, and 10 H stored in a working Register. Each of these numbers is consecutively subtracted from the data to yield the 3 digits of BCD data which are stored in memory (RAM) in the format shown in Figure 14b. These digits can then be combined into the desired format.

Some applications for the LD130 will now be discussed. A simple application of the ratio measurement feature of the LD130 would be as an angular or linear position measuring instrument. The linear potentiometer and two resistors, shown in Figure 15, are the only additional components needed for this measurement and serve both to provide the position to voltage conversion (0 to 100%) and the tracking reference voltage. Applying the voltage divisions from these resistors to the equation for COUNT

Gives $V_{REF} = V_1 \frac{2R}{4R} = \frac{V_1}{2}$ and,

$V_{IN} = V_1 \frac{R}{4R} X$ where X is the pot coefficient (ranging from 0 to 1)

then $\text{COUNT} = \frac{V_{IN}}{V_{REF}}(2000) = \frac{V_1(R/4R)X}{V_1(2R/4R)} 2000$

which gives COUNT = 1000X

Scaling to degrees, centimeters, or any specific measurement unit simply involves solving for the appropriate divider resistors using the equation for COUNT.

A digital Thermometer, reading in either °F or 0.1 °C, can be constructed by adding the circuit shown in Figure 23 to a basic LD130 DPM. This circuit converts temperature to a voltage by using the temperature dependent forward voltage of a PN junction as the sensing element. The change in this voltage is typically −2.3 mV/°C at room temperature and can be suitable linear if the PN junction is biased with a constant current much greater than the reverse saturation current.

Since the diode will have a finite voltage at either a Celsius or Farenheit Zero, this voltage component must be subtracted out. Figure 16 shows the temperature sensing diode

(Base-Emitter of a 2N2222) biased with a 0 T.C. Current regulator diode (CRO33). Zeroing is achieved by summing the currents from the CR033 and E506 diodes at the wiper of a potentiometer.

The scaling for either Celsius or Farenheit is achieved by adjusting V_{REF} according to equation for COUNT.

$$\text{Count} = 2000 \frac{V_{IN}}{V_{REF}}$$

$$\Delta\text{Count} = 2000 \frac{\Delta V_{IN}}{V_{REF}}$$

$$\text{for } \Delta T = 100^\circ\text{C} \quad \Delta V_{IN} \cong -230 \text{ mV}$$

$$\Delta T = 1000^\circ\text{F} \quad \Delta V_{IN} \cong -1.278 \text{ V}$$

Thus

$$V_{REF} = 0.46 \text{ V for } ^\circ\text{C}$$

$$= 2.5 \text{ V for } ^\circ\text{F}$$

The fact that the forward voltage decreases with temperature requires that the sense of the LD130 sign bit be inverted ("0" for +, "1" for −).

The LD130 offers substantial advantages in terms of cost, power, simplicity, and versatility to the A/D market. This chip offers simple interfacing the μP systems requiring few additional components and a simple software solution.

INTERFACE AGE thanks Gary Grandbois of Siliconix, the author of Application Note AN 76-5 "Applying the LD130" from which much of this material is derived. Further thanks go to Bryon McCullough, sales engineer for Siliconix who provided the application assistance without which the article could not have been written.
Next month we shall cover a collection of I/Cs which should be of interest. —*R.H. Edelson*

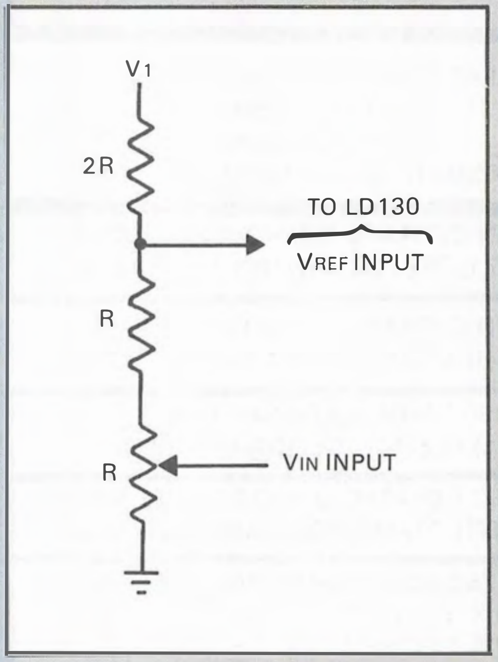


Figure 15.

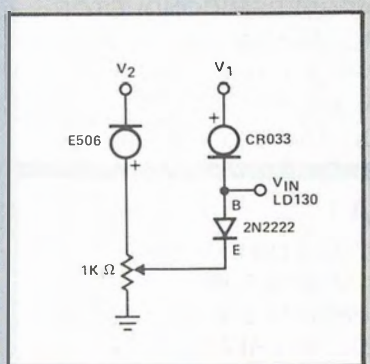
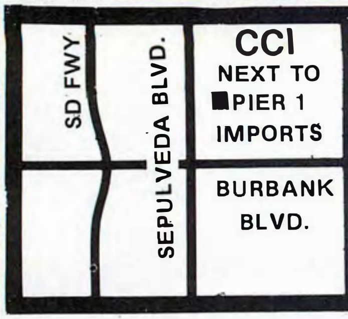


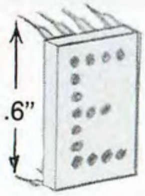
Figure 16. Temperature to Voltage Converter



COMPUTER COMPONENTS

5848 SEPULVEDA BLVD.
VAN NUYS, CALIF. 91411
(213) 786-7411

HOURS
TUES. - FRI.
10 A.M to 9 P.M.
SAT. 9 to 9
SUN. 10 to 6
CLOSED
MONDAY



HEXIDECIMAL DISPLAY
HP 5082—7340

$7.00(1) $6.00(10) $5.00(100)

!! FACTORY PRIME INTEL 2708'S
1,024 BYTE EPROM
UV ERASEABLE
$65.00(1) $50.00(10) $40.00(100)

74150 IN CERAMIC DIP HOUSE NUMBER
$1.00(1) 75¢ (10) 60+(100) 45¢ (1000)

74181 IN CERAMIC DIP HOUSE NUMBER
$1.25(1) $1.00(10) 70¢ (100) 50¢ (1000)

74190 PRIME NATIONAL PARTS
90¢ (1) 80¢ (10) 70¢ (100) 60¢ (1000)

AMS 6002 CERAMIC DYNAMIC RAM 1Kx1
$1.00(1) 70¢ (10) 50¢ (100) 35¢ (1K)

DB25's (S)$3.00 (P)$2.00 SHELLS $1.00
IMSAI 100 PIN CONNECTORS $5.00 ea.
DIP SW'S 8 POSN $2.50 7 POSN $2.25

20K SLIDE POTS (STACKPOLE 114)
50¢ (1) 40¢ (10) 35¢ (100)

HPROM 1024 BY HARRIS SEMICONDUCTOR
256x4 FIELD PROGRAMMABLE ROM
THESE HAVE BEEN PROGRAMMED BUT MAY HAVE HEAVY VIRGIN AREAS
25¢ EA.(1-1000) 15¢ EA.(1000 up)

IMSAI

I8080-1K MICROPROCESSOR KITS
RAM 4A-4 MEMORY MODULE KITS
PIC 8 PRIORITY INTERRUPT & RTC KITS
SIO 2-1 SERIAL RS232 INTERFACE KITS
PIO 4-4 4 PARRALLEL INPUT/OUTPUT PORTS
EXP 22 22SLOT MOTHERBOARD

TELETYPES

* KSR 33 $450.
* KSR 35'S FROM $500 to $1200
* ASR 33'S FROM $600 to $1000
* ASR 35'S FROM $750 to $2500

"D" CELL BATTERY HOLDER

65¢ (1) 50¢ (10) 40¢ (100)

TO-3 HEAT SINK

50¢ (1) 40¢ (10) 35¢ (100)

POLYMORPHIC SYSTEMS

POLY88 MICROPROCESSOR
A COMPACT, POWERFUL COMPUTER

POLY I/O PROTOBOARD
FULL BUS INTERFACE LOGIC PLUS ROOM FOR 59-16 PIN WW SOCKETS

VTI/64 VIDEO DISPLAY DRIVER
128x48 GRAPHICS & KEYBOARD INPUT

CROMEMCO

BYTESAVER 8K EPROM USES 2708
D+7AIO D/A A/D 5 MICROSECOND CONVERSION
TV DAZZLER COLOR TV CONTROLLER
JOYSTICKS FOR PROPORTIONAL CONTROL

Please add $2.00 Postage and Handling on orders under $25.00. Sent UPS. Add $4.50 Postage and Handling on Orders under $50.00. Sent Post Office.

$10.00 MINIMUM ORDER
B of A CARD * COD * PHONE ORDERS

California Residents Add 6% Sales Tax

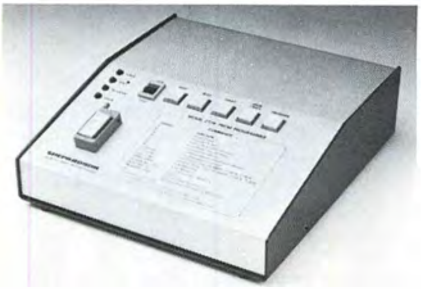
CIRCLE INQUIRY NO. 45

New Products

COMPONENTS

Low Cost PROM Programmer for 2704 & 2708 PROM's

Model 2708 PROM programmer, is an intelligent programmer that will program all major manufacturer's 2704 and 2708 PROM's.



The Model 2708 provides an RS232 and 20mA current loop interface as a standard feature. This allows the programmer to be used in conjunction with all printing and CRT type terminals, a paper tape reader/punch, or as a peripheral to the user's computer at band rates up to 600.

The list price of $850 makes it the lowest cost, highest performance programmer available. Because it has it's own internal microprocessor, the Model 2708 provides many features and capabilities not available in other programmers.

1. Full editing capability allows the user to move, alter, and store data in the programmer's buffer memory in the user's choice of number base, binary, octal, decimal, or hexadecimal.
2. A printing terminal can provide the user with hard copy of the PROM data.
3. A paper tape reader can be used to input the desired data, and a paper tape can be generated to store the PROM data for later use and/or modification. The paper tape may be in BNPF, BHLF, binary, or ASCII hexadecimal format.
4. The programmer will automatically adapt to the terminal baud rate up to 600 baud.
5. The user can modify both the pulse width and the number of loops in programming a PROM, when used in conjunction with a terminal.

The versatility and low cost of the Shepardson Model 2708 make it a most attractive product to the user of 2704's and 2708's.

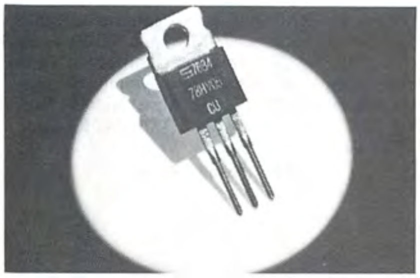
For further information contact R.C. Shepardson, Microsystems Inc., 20823 Stevens Creek Blvd., Bldg. C4-H, Cupertino, CA 95014; (408) 257-9900.

CIRCLE INQUIRY NO. 90

7800-Type Voltage Regulators from Signetics Feature Breakdown Rating Guaranteed to 60 Volts

An improved series of 7800-type high voltage regulators with a breakdown voltage rating 50% higher than competitive units is now available from Signetics.

The new devices, designated 78HV00, are 1 amp, three-terminal positive voltage regulators with a guaranteed input breakdown of 60 volts. The devices directly replace standard 40-volt versions of 7800 regulators. The higher rating can improve system reliability and permits larger transients and AC line fluctuations without system damage.



The 78HV00 units are the strongest 7800-type regulators available. With thermal shutdown and output transistor safe-area compensation incorporated in the monolithic package, they are virtually indestructible in standard circuit applications. Output current with adequate heat sinking is typically in excess of the specified 1 amp rating.

The 78HV00 regulators require no external components in fixed voltage regulator applications and offer internal thermal overload protection. Units in the line are available with output voltage ratings of 5, 6, 8, 12, 15, 18 and 24 volts in either TO-220 or TO-3 packages.

They are ideal for a wide range of applications, including local, on-card regulation for eliminating power distribution problems associated with single-point regulation. The 78HV00 devices can also be used with external components to obtain adjustable output voltages and currents, or as the power pass element in precision regulators.

Operating junction temperature ranges for the new units are available to −55 to +150 degrees C.

The 78HV00 high voltage regulators are available from stock through Signetics and its authorized distributors. As an example of pricing, 78HV05CU rated for 5 volts output in a TO-220 package is $1.56 in quantities, of 100-999; 178HV24CDA rated for 24 volts in a TO-3 package is $2.26 in the same quantities.

For further information contact Signetics, 811 East Argues Avenue, Sunnyvale, CA 94086; (408) 739-7700.

CIRCLE INQUIRY NO. 91

Priority Encoders

Two eight-line-to-three-line low-power Schottky priority encoders are now available from Advanced Micro Devices.

The Am25LS148 performs priority decoding from eight inputs and provides a binary weighted code of the priority order of the inputs. This device is available in a 16-pin package and offers standard totem-pole outputs.

The Am25LS2513 is a gated three-state output version of the Am25LS148 in a 20-pin package. This device is particularly useful in the design of priority interrupt systems.

Both devices are available in molded and hermetic DIP and ceramic flat packages. Prices for the Am25LS148 start at $1.94 in 100-piece lots.

For further information contact Advanced Micro Devices, Inc., 901 Thompson Place, Sunnyvale, CA 94086; (408) 732-2400.

CIRCLE INQUIRY NO. 92

Keyswitches

Build custom keyboards in minutes with the KBM series keyswitch. The model KBM is a low-cost, long life keyboard switch ideal for use in CRT terminals, data entry devices, hex keypads, touch tone encoders, etc. Semi-sealed construction means immunity to environmental contamination and the gold crosspoint contacts insure reliable, low bounce switching for more than 10 million operations. Contacts are rated at 12 volts, 1ma, with less than 100 milliohms contact resistance. Mounting through an easily fabricated metal panel insures precise keycap alignment, and the rugged 1/16" terminals make hardwired prototypes a breeze.

A variety of keycap sizes, shapes, colors, and legends are available.

For further information contact George Risk Industries, Inc., GRI Plaza, Kimball, Nebraska 69145. Telephone (308) 235-4645. TWX 910-620-9040.

CIRCLE INQUIRY NO. 93

Smallest Coded DIP Switch

"MICRO-DIP," a ten position coded switch, occupies only one half of a 14-pin DIP socket and it can be mounted/connected directly to circuit boards by hand or flow soldering.

Switch settings can be made in either direction with a screwdriver. Numbers on top of switch indicate detent positions. MICRO-DIP can handle loads up to 100 milliamps at 5VDC and operate in a temperature range of −10° to +60°C.

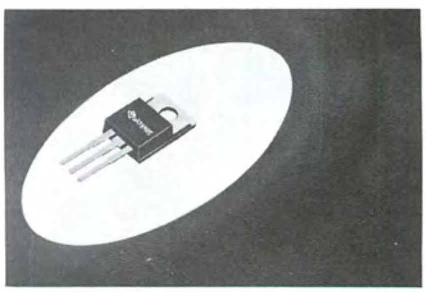
MICRO-DIP contacts are gold-plated. The .380" x .400" x .240" high glass filled epoxy housing provides a dust seal. Available codes include BCD complement, 1 or 2 pole 2 throw and 1-pole 5-position repeating in design. Priced under $1.00 in 10K quantities; 6 week delivery for prototype units.

For further information contact EECO, 1441 East Chestnut Av., Santa Ana, CA 92701; Phone "Switch Products" (714) 835-6000.

CIRCLE INQUIRY NO. 94

TI Regulators Keep Output Voltages Constant

Texas Instruments Incorporated has just sourced the very popular μA78M and μA79M series of fixed voltage regulators.



The μA78M series are positive types, including those with nominal output voltages of 5V, 6V, BV, 12V, 15V, 20V and 24V. They are

available in the TO-220 KC package with an operating temperature range of 0°C. to 70°C.

The μA79M series are negative types in the same voltage ranges. Both series are available in the TO-220 KC package and can deliver up to 500 milliamperes of output current.

The μA78M and μA79M series are interchangeable to the Fairchild series with the same designation.

Prices for 100 units are as follows: μA78M series: $1.09 each; μA79M series: $1.09 each.

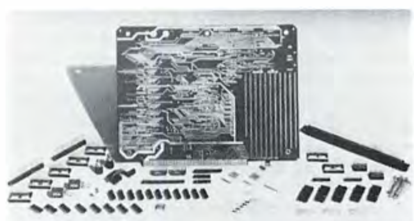
For further information contact Texas Instruments, Inc., Inquiry Answering Service, P.O. Box 5012, M/S 308, (Attn: Bulletin Nos. DL-S 7612403 and DL-S 7612405, June 1976), Dallas, TX 75222.

CIRCLE INQUIRY NO. 95

EQUIPMENT

UC 1800 Microcomputer

The UC 1800 is completely assembled and tested. As such, it has important design cost saving advantages to industrial users contemplating the use of microprocessors in their products.



The comprehensive instruction manual, simple straightforward software instructions, self-contained keyboard, and four digit hexadecimal display provide a package which promotes rapid training and system development.

All users will benefit from the growth potential incorporated in the UC 1800. External bus access allows future connection to a host of peripheral devices and add-on memory which can provide full mini-level computer power.

The user of the OEM version will find such features as full military temperature range, low power CMOS, single 3 to 15 volt supply and TTL compatibility are decided advantages in a wide range of product applications.

The outstanding features are: Low cost; Built-in keyboard programming; Digital (hexadecimal) display for address, memory contents, and I/O port; Front panel control of Interrupt, DMA, I/O Flag; 256 byte RAM expandable to 65.5 K bytes RAM or ROM externally; Low power consumption; Special circuit saves memory content when unit is turned off; Single power supply; Parallel and serial I/O data line capability; Available as single PC board microcomputer with or without on-board power supply for OEM applications.

The comprehensive documentation package of 230 + pages included with the UC 1800 provides extensive tutorial material with training aids on the theory and operation of digital computers. Both hardware and software are covered. Step by step instructions are given on programming. Programs for machine checkout and applications program examples are included.

Software included with the UC 1800 contains KEYBUG™, a keyboard handler and debug program which provides powerful software tools for generating and debugging your own programs.

A software library is available which contains diagnostics, useful macro building block routines, and games.

Option 001 — Automatically recharged internal battery. Allows program memory to operate for up to four hours after power failure, $22.50.

Option 002 — 120/230 VAC 50-500 Hz input power, *specify which voltage is desired*, $15.00.

Also available in kit form, $395.00.

For further information contact Infinite Incorporated, 1924 Waverly Place, Melbourne, FL 32901; (305) 724-1588.

CIRCLE INQUIRY NO. 96

Designer Evaluation Kit for Signetics 8X300 Bipolar Microprocessor

An evaluation kit that allows design engineers to evaluate the Signetics 8X300 bipolar microprocessor for planned applications is now available from Signetics.

The single-board evaluation kit includes all the elements a designer needs to judge the suitability of the 8X300 for system applications. The unit includes the 250-nanosecond bipolar microprocessor CPU, four input/output (I/O) ports for interfacing external devices, 256 bytes of working data storage. Programmable read only memory (PROM) devices in the kit are preprogrammed with input/output control logic, random access memory (RAM) control, and RAM diagnostic programs.

The microprocessor kit, designated 8X-300KT100SK, is intended to familiarize design engineers with the performance advantages of the 8X300, particularly for control and data movement applications.

The 8X300 features a 13-bit address bus for sellcting instructions from program storage, and a separate input bus for entering 16-bit instruction words. With the 8X300, the entire process of data input, shifting, processing and output is accomplished in a single instruction cycle of 250 nanoseconds.

The board layout of the kit allows for variations and expansions of the basic design. All signals, for example, can be conveniently transferred off the board, and a wire-wrap area is provided for additions to the board circuitry. Additions may include memory, interfaces or special circuits for specific user requirements.

The 8X300 evaluation kits are available from stock through Signetics and its authorized distributors. Unit price of the kit is $299.00.

For further information contact Signetics, 811 East Arques Ave., Sunnyvale, CA 94086; (404) 739-7700.

CIRCLE INQUIRY NO. 97

Communications Processor

The 40 Series Communications Processor is designed to perform such functions as data concentration, channel contention, message routing, polling control, speed and code conversion, protocol conversion, and voice response. The program is executed in nonvolatile Programmable Read Only Memory (PROM), and need not be "reloaded."

The basic system is self-contained on a single card Central Control Module, including a LED display and 10-position function switch for diagnostic test or system monitoring. Modular construction permits cost-effective single-channel or multi-channel configurations, with CPU, RAM buffer storage, PROM control firmware, synchronous and asynchronous communication interfaces, voice synthesizer channel modules and optional operator's console consisting of CRT and keypad.

Additional features include an Auto-Restart Timer and redundant common logic and power supplies for maximum reliability. The Voice Synthesizer produces clear, natural, male or female speech.

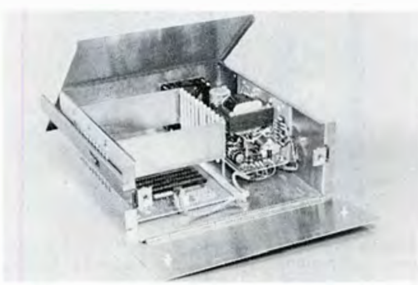
Normally supplied in turnkey communications controller systems complete with firmware, the 40 Series is also available to OEM's with a complete Program Development System to facilitate firmware development. Prices start at $1800, with delivery of 45 days ARO.

For further information contact Roger Evans, Micom Systems Inc., 9551 Irondale Avenue, Chatsworth, CA 91311; (213) 882-6890, TWX 910-494-4910.

CIRCLE INQUIRY NO. 98

μC Hardware for the OEM

Supporting M6800-based equipment development, are chassis, card cages and a power supply for the MICROMODULE™ products. Two card cages are available; one with 5 card slots, the other with 10 card slots. Both card cages have mother boards that are pin compatible with the MICROMODULES and all of the EXORciser modules.



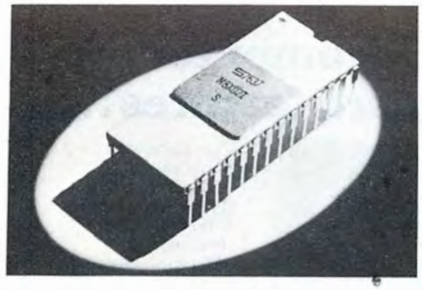
The cages can be mounted on five different axes. The unit price of a 10-card cage is $147.00; a 5-card cage sells for $98.00.

For further information contact Motorola Semiconductor, Inc., P.O. Box 20912, Phoenix, AZ 85036; (602) 244-6900.

CIRCLE INQUIRY NO. 99

Control Store Sequencer Simplifies Bipolar Microprocessor Design

The random logic needed to implement microprogram sequencing in bipolar microprocessors can now be replaced by a new control store sequencer from Signetics. Designated 8X02, the new unit forms a powerful microprogrammed control section for computers, controllers or sequenced logic elements, when combined with standard Read Only Memory (ROM) or Programmable ROM (PROM).



The Signetics 8X02 is a low-power Schottky LSI device designed for use in high-performance microprogrammed systems. The basic function of the 8X02 is to control the fetch sequence of microinstructions. The unit is capable of addressing up to 1k words of microprogram, expandable to any microprogram size by conventional paging techniques. External page registers, when they are provided, can be controlled entirely or partially by the microprogram.

The 1k addressing capability is not available in other units, nor is the 8X02 subroutine nesting capability. The Signetics unit has a 1k addressing capability and a 8X02 subroutine nesting capability. A four-way stack register file and offers conditional nesting for branching to a subroutine or for automatic interrupt handling.

The 8X02 is easy to use, requiring just three control lines for the three-bit command code, versus from 9 to 18 lines for alternate units. Control instructions include Increment, Test and Skip, and Conditional branch to subroutine. The 8X02 uses a popular 28-pin plastic package.

The low-power Schottky process employed for the sequencer features good performance for both medium-scale computer applications and controllers. Chief applications are expected to be in central processing units (chiefly minicomputers); peripheral controllers such as high speed magnetic disc storage units, floppy discs and tape drives; vector generators for all types of CRT display terminals; and simple step controllers of various kinds. The 8X02 can be used for any system, however, that requires sequencing of instructions.

Cycle time for the 8K02 is 80 nanoseconds. The unit operates with a +5-volt power source with power consumption of 1000 milliwatts. The unit is totally compatible with all bipolar TTL logic elements.

The control sequencer architecture is shown in the schematic. The address register consists of ten D-type edge-triggered flip flops with a common clock. The address register can be loaded with different address sources under control of the three address control lines and one test input line.

The four-register stack input line with a two-bit stack pointer respond automatically to operations requiring a PUSH (write to stack register file) or POP (read stack register file). The file is organized as a four-word-by-ten bit matrix and operates on a last in first out basis. The stack pointer operates as an up/down counter.

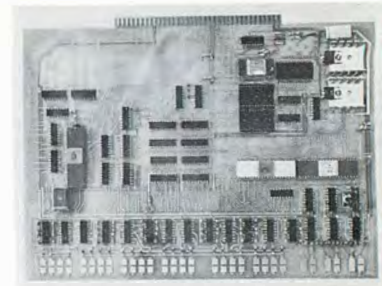
The N8X02XL Control Store Sequencer is available from stock through Signetics and its authorized distributors. Pricing is about $19.45 in 100 quantities.

For further information contact Signetics, 811 East Arques Avenue, Sunnyvale, CA 94086; (408) 739-7700.

CIRCLE INQUIRY NO. 100

Datac 1000 Computer, Controller and Tutorial Card

The Datac 1000 card is available in two configurations: tutorial and populated. The tutorial has the folowing features: The MOS Technology 6502 microprocessor chip; one page of RAM (256 bytes); one-bit output latch; 8 data, 8 address and 9 control touch pads (our method of entering data with elegance and simplicity); 8 data and 8 address LEDs; single-cycle operation.



In the fully populated form the Datac 1000 is a computer with the following features: full 16-bit addressing; total of 1K bytes of RAM (8x-2111); sockets for 2K of UV ROM (2x2708) for use as a dedicated controller; MOS Technology 6530-004 teletype I/O monitor (in ROM) plus 64 bytes of RAM plus 8-bit I/O port; complete interface to teletype current loop or EIA RS 232; high-speed cassette interface using the MC6850 ACIA, for storing programs; 6820/6520 PIA providing two 8-bit parallel I/O ports; fully buffered tri-state busses; power-on reset or restart; on-board address decoding for expansion; ribbon connector for I/O and edge connector for expansion.

The board is designed with capability for full expansion as described above, and for populating to a lesser degree for use as a controller. The Datac 1000 E fully populated card (less EROM) is available assembled and tested with power supply and a manual in single quantities for $345.00. Availability is stock to 60 days.

For further information contact DATAC Engineering, P.O. Box 406, Southampton, PA 18966; (609) 854-7852.

CIRCLE INQUIRY NO. 101

Precision Forty-Channel CB/RF Generator Achieves Crystal Accuracy in Low-Cost Unit

The RF Generator designated the Model 256 for 40 channel CB tranceiver service incor-

porates features of particular benefit to CB service technicians. Five-band frequency tuning covers channels 1 through 40 on an expanded tuning range for easy, precise channel selection. Frequencies of 100 kHz through 16 MHz are covered on the other four bands to provide all IF requirements including: 455 kHz, 10.7 MHz, and any other, current or future.

Precision frequency selection is accomplished by connecting the counter output jack to a frequency counter for continuous monitoring. By use of the counter output in conjunction with a frequency counter crystal-controlled accuracy is available without the usual high cost.

A calibrated/attenuated output control provides RF signal output of 100,000 μV down to less than 1 μV for receiver sensitivity checks. The attenuated output is variable in 20 dB steps and by a 20 dB continuously-variable control calibrated in microvolts.

Internal modulation at a frequency of 1kHz is variable from 0 to 100%, calibrated at 30%. Provision is also made for use of external modulation at frequencies from 20 Hz to 10kHz through front-panel out/in jacks. When the Audio Output function is selected a 1 kHz audio signal

is available at these same front-panel out/in jacks.

The Hickok Model 256 CB/RF Generator is available through Hickok distributors. Suggested retail price is $199.00.

For further information contact Marketing Services Department, Hickok Electrical Instrument Company, 10514 Dupont Avenue, Cleveland, Ohio 44108.

CIRCLE INQUIRY NO. 102

Teletype Mobil Cart

The TTS Cart attachment adds mobility to any Teletype terminal. It is affixed to or removed from any Teletype in 30 seconds. It offers hand truck-like leverage and easy movement over door jambs, carpet edges or even stairways. When the terminal and stand are in the upright operating position the wheels remain suspended above the floor to insure fixed installation stability.

The mobilized terminal may be rolled to any desirable location with ease.

Made of cold-drawn 1" steel tubing, high-tensile steel alloy spring clamps, 8" x 1½" ball-bearing wheels. Size: height 38"; width 32". Weight: 12 pounds. Single unit price: $59.50.

For further information contact TTS, 2928 Nebraska Ave., Santa Monica, CA 90404; (213) 829-2611.

CIRCLE INQUIRY NO. 103

Electronic 'Game-Playing' Organ

A computer-driven electronic organ that either plays your favorite song or teaches you how to play it has been developed.

The organ is one of three electronic "game-playing" devices that can be operated simultaneously by a single computer — in this case, a Computer Automation millicomputer, one of the smallest on the market.

The organ is controlled by a Computer Automation NAKED MILLI LSI-3/05, a compact, general-purpose machine priced as low as $395.

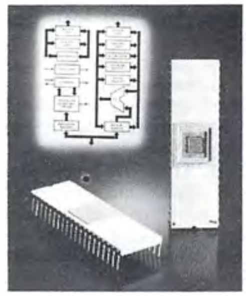
In the case of the organ-player, the NAKED MILLI is literally easy enough for a child to operate. For example, in order to learn how to play a particular song, the operator presses a start button and lights are illuminated over the organ keys to be pressed, in timed sequence. An automatic recording and playback feature enables the user to hear the notes he has played. He also has the option of simply playing any of the pre-recorded selections by pressing the appropriate button.

For further information contact Computer Automation, 18651 Von Karman, Irvine, CA 92713; (714) 833-8830; TWX 910-595-1767

CIRCLE INQUIRY NO. 104

"SC/MP-II" Microprocessor — Higher Speed, Lower Power Consumption, Single Voltage Source

Samples are now available of a new N-channel MOS version of the "SC/MP" 8-bit single-chip microprocessor that is twice as fast and which uses only one-fourth as much power as the P-channel version.



The "SC/MP-II" microprocessor has all of the features of the older version while offering several advantages over the P-channel device.

Power consumption of 'SC/MP-II' is less than 200 milliwatts, considerably lower than the approximately 800 milliwatts consumed by the earlier version and the lowest power consumption of any N-channel MOS processor on the market today. Another significant improvement is that they have eliminated the need for two power sources. The 'SC/MP-II' chip needs only a single source of +5 volts for operation, which is a great improvement over the first model which required two power sources — a +5 volt and a −7 volt supply.

The speed of "SC/MP-II" is twice that of the P-channel model. The new version takes one microsecond to complete a microcycle, and typical instruction execution time is 5 microseconds.

The "SC/MP-II" is fully compatible with its predecessor in terms of pin configuration, object code, and software, and with slight modifications to the crystal frequencies, it will be compatible with all of the "SC/MP" support equipment, such as the "SC/MP KIT" in the U.S.A., the "SC/MP INTROKIT" in Europe, the "SC/MP LCDS" (Low-Cost Development System), and the "SC/MP Keyboard Kit" which eliminates the need for a teletype machine.

The clock oscillator, which is located on the "SC/MP-II" chip, is designed to use very inexpensive television-type crystals of 3.58 or 4.0 megahertz. As an alternative to a crystal, the user may drive the clock with a standard TTL (transistor-transistor logic) timing system. In addition to the clock, all of the inputs and outputs are compatible with TTL devices and can also be easily interfaced with MOS and CMOS circuitry.

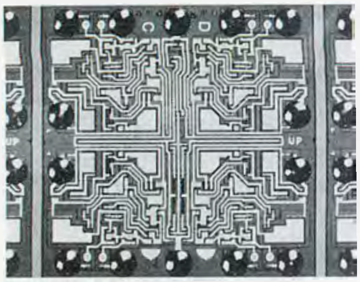
Sample quantities of the "SC/MP-II" microprocessor are immediately available from the factory. The price for a single sample is $17.76, and production quantities of more than 2,000 will sell for approximately the same prices as the P-channel "SC/MP." Prices will be considerably lower in 1977 because we will then be able to offer the 'SC/MP-II' in a plastic package.

For further information contact National Semiconductor, 2900 Semiconductor Drive, Santa Clara, CA 95051; (408) 737-5000, TWX 910-339-9240.

CIRCLE INQUIRY NO. 105

Quad OP-AMP Flip-chip

The popular MC3503 type quad operational amplifier is now available in flip-chip form as well as in conventional chip form and a variety of plastic and hermetic packages. The flip-chip consists of a silicon chip with solder bumps (90-10 solder on a chrome-copper-gold base) on the geometry surface to provide easy mechanical mounting and electrical connection.



Electrically the MCCF3503/3403/3303 offer much better amplifier matching than four single operational amplifiers. Each amplifier has characteristics similar to that of the MC1741 and in addition is designed with a Class AB output stage which minimizes cross-over distortion.

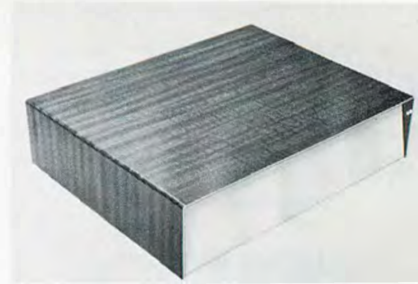
The flip-chip is offered at the following prices (in quantities of 100-900). MCCF3303 ($-40°C$ to $+85°C$) \$2.25; MCCF3403 (0 to 70°C) \$2.50; MCCF3503 ($-55°C$ to $+125°C$) \$6.75.

For further information contact Linear Marketing (602) 962-2122 or the Technical Information Center, Motorola Semiconductor Products, Inc., Box 20294, Phoenix, Arizona 85036.

CIRCLE INQUIRY NO. 106

Cases Enhance Custom-Designed Instruments

This line of economical enclosures, in two sizes and numerous attractive colors and finishes, gives desk-top instruments an appearance contemporary with sophisticated electronic components and systems. Designated Cono-Cases by Vector Electronic Company, the WA series enclosures incorporate a ten-degree sloped front panel and an optional smoked-plastic facing for behind-panel indicators. A recessed rear panel protects input/output connectors.



The enclosures, assembled from two interlinked channels, allow easy access to circuits, accessories and wiring. The lower section forms a chassis integrated with front and rear panels. Elongated holes in the bottom and rear panel provide superior convection cooling. The upper section serves as top and side panels. The WA1 enclosures are 11 inches wide by 8 inches deep by 4 inches high, giving 307 cubic inches of circuit space.

The WA2 enclosures are 14 inches wide by 11 inches deep by 4 inches high, providing a 560 cubic inch working volume. Construction of 0.062 inch (14 gauge) aluminum insures adequate support for transformers, heat sinks and other heavy components.

Cono-Cases are available with clear anodize satin finish, or with blue or walnut grained vinyl on the cover. Other colors available in anodize, vinyl or paint on request.

The WA Series enclosures are priced from \$12.95 to \$19.70, depending on model and finish. They are available off-the-shelf from Vector and will be available through the firm's distributors throughout the United States and Canada.

For further information contact Vector Electronic Company, 12460 Gladstone Ave., Sylmar, CA 91342; (213) 365-9661; TWX (910) 496-1539.

CIRCLE INQUIRY NO. 107

Low Cost Logic State Analyzer

A logic state analyzer, priced at \$272, including shipping and handling, places data domain analysis within the reach of smaller companies, educational institutions and hobbyists who previously could not afford the luxury of this useful development and troubleshooting technique.

The instrument, dubbed the Model 100A, is designed to operate with an ordinary oscilloscope and incorporates many of the features found on much more expensive analyzers including: a 16-word truth table display of ONE's and ZERO's; eight input channels with eight corresponding trigger word switches which can individually be set to "1," "0," or "X" (don't care); an internal data memory for post-trigger data collection; hexadecimal and octal formats; and both static and dynamic display presentations. These features permit the Model 100A to be used in typical logic state analysis applications such as tracing computer program flow, examining the contents of ROMs and other memories, checking counter and register operations, observing I/O sequences, and monitoring micromemory address steps.

The Model 100A can be used directly with a variety of logic families including TTL, Schottky, MOS, CMOS, and DTL. Connection to the

system-under-test is accomplished using a color-coded flat ribbon cable terminated in gold plated "universal" pin connectors. These connectors can be used with wire-wrap pins, IC clips, data grabbers, or ball clips. An input data rate in excess of 8 Megabytes/second assures compatibility with most microprocessors and special-purpose digital systems.

The instrument comes with a 100-page manual which includes the theory of operation of the unit and the analysis procedures for seven common microprocessors. Also included are sections describing production testing, field service, and educational applications. Delivery is stock to 30 days.

For further information contact Paratronics, Inc., 150 Tait Avenue, Los Gatos, CA, 95030; (408) 354-7766.

CIRCLE INQUIRY NO. 108

Math Book Bonus with Purchase of Calculator

Anyone who purchases a TI-30 calculator soon will receive a free bonus — a 200-page book entitled "The Great International Math on Keys Book" that can help high schoolers with their math studies and others to utilize math in everyday life applications.

At a suggested retail price of $24.95, the kit is designed for students and others who want to make effective use of the calculator's capabilities in both classroom and home. It will be available through a wide variety of retailers.

For high school students, the new math kit will provide an opportunity for supplemental studies and exercises in algebra, trigonometry, physics, chemistry, probability and statistics. Ways to use the calculator to perform conversions are also included in the book since the metric system is now being adopted in the United States. Students are learning currently to measure gallons of gas in liters and distances in meters instead of feet or miles.

Students and others can also use the book and calculator on home management problems. For example, there are sample problems which show how easy it is to determine the amount of paint needed for a room area, quantity of fencing for yards or cement for patios. There are also hints for smart supermarket shopping including finding unit costs and managing a budget at the market.

Numerous other exercises and hints cover how to use the calculator in personal business and finance applications. Included are such things as credit card purchases, figuring mortgage points, interest cost, depreciation and foreign currency conversions.

The closing part of the book introduces the users to some entertaining puzzles and games they can enjoy with the calculator. Some examples are calculating biorythms, doing crossword puzzles and just having fun with numbers.

For further information contact Texas Instruments, Incorporated, P.O. Box 5012, Dallas, TX 75222; (214) 238-2481.

CIRCLE INQUIRY NO. 109

Slim Profile

The UNIVUE keyboard and instrumentation enclosure is designed for low profile instrumentation and computer data entry applications. Special design features include a welded .062 inch steel body of two piece construction for strength and an outside removable, flush mounting aluminum panel.

Overall size is 24"x12"x3", with front panel size of 23"x8". The UNIVUE body is coated with light gray lacquer type primer/surfacer to allow virtually any type paint to be used in the application of final finish. The UNIVUE is shipped complete with aluminum panel, rubber feet, and mounting hardware. Shipping weight is 17 lbs., with delivery from stock. 1-24 price is $32.95.

For further information contact Advanced Data Sciences, P.O. Drawer 1147, Marion OH 43302; (614) 382-7917.

CIRCLE INQUIRY NO. 110

Five New Convection-Cooled Switching Power Supply Models

Powertec now offers five new models of the industry's highest power density, off-the-shelf, convection-cooled switching regulated power supply series. Packaging density is achieved as a result of the mechanical configuration, and through utilization of a 40kHz switching rate - also the industry's first.

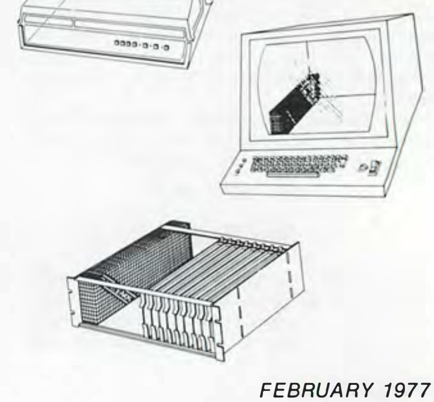
The new series, designated 9E, is a 250-watt switching regulated power supply in a 2.25" x 4.94" x 15.0", 6 lb. 3 oz. package yielding 1.4w/in³. This "low profile" package is provided in response to an extensive survey performed late last year by Powertec of a broad base of power supply users. This profile allows it to fit crosswise in a standard 5.25" Retma rack while using only 2.25" of rack depth or it may be mounted to the cabinet alongside, above, below, or behind the rack drawers. Additional applications are alongside CRT's in video display terminals, in the side panels of consoles, in card cages, in briefcase mounted to portable test equipment, or any number of other applications that an innovative engineer can imagine.

The input voltage range for this supply is 115/230Vac +10/−20% with a 20mS hold-up allowing it to work under abnormal line conditions. Under "normal" ±10% line conditions, this unit will operate from 110, 115, 208, 220, or 230Vac nominal line voltages. Ripple and noise is 50mV p-p, regulation is 2mV for a full 30% line change and 0.2% for a full load change and transient response is ≤400/μs for recovery to 0.1% with a maximum deviation of 250mV for a 50% load change. Overall efficience is 80% minimum.

Standard features include remote sensing and voltage programming as well as OVP and logic inhibit. Options include factory installed "master-slaving" that does not require matching masters with slaves and allows masters and slaves to be interchangeable. A power-fail logic output is also optional.

The control circuitry is all new and provides many protective features not previously offered by any supplier. As with most other switching regulators, protection against over-voltage, over-current, over-temperature as well as inadequate line voltage is provided. In addition to these there are 5 additional proprietary internal features that protect the supply against operating conditions that would prove fatal to many other switchers but ensure safe operation at 40kHz in the 9E5-50C-17. These include circuits that keep transistor collector currents and transformer flux densities balanced and within design limits under all normal and abnormal operation conditions. To demonstrate inherent reliability, 10 units are presently undergoing perpetual life test. Also, every production unit is subjected to a 24-hour burn-in. These burn-ins include repeated line, load, short-circuit, and thermal cycling as well as full load operation at a 40°C ambient.

This supply in unit quantities sells for $395, which makes it one of the least expensive



switching supplies available in this power range. Units have been provided to Underwriters' Laboratories for investigation and recognition per UL478 & UL114. Units are available for demonstration and evaluation now with production quantities available in October.

For complete details and specifications on the 9E Series, send for the new "Contortionist" brochure.

For further information contact Powertec, Inc., 9168 De Soto Avenue, Chatsworth, CA 91311; (213) 882-0004.

CIRCLE INQUIRY NO. 111

Prompting Desktop Programmable Calculator

A prompting programmable desktop calculator featuring a display that communicates with the user has been introduced by Texas Instruments Incorporated. The desktop SR-60 includes a printer, magnetic card reader and more individual function keys than any other calculator.

With its "prompting" display, an SR-60 user can run alphanumeric programs which request information through the 20-character display at successive stages in a problem. The calculator then waits for a response before continuing with problem solving. This "dialogue" allows even a novice to work with complicated problems immediately.

The SR-60 is designed for both business and technical operations. In business, it it capable of many functions, including financial analysis, long-term forecasting, and payroll. In technology, its 46 exposed scientific functions are immediately available for calculations, and 480 program steps, expandable to 1920 with an optional module, are available for complex programming.

Although it has high programming capability, the SR-60 can also be operated easily as a general purpose calculator. Its left-to-right algebraic entry and 9 levels of parentheses allow problems to be entered exactly as a user would say them. Answers can then be displayed, printed, or both — at the user's option.

The wide range of keyboard functions includes trig functions, hyperbolics, powers, roots, logs, Δ%, factorials and many other mathematical functions. Degrees of accuracy can be precisely controlled to provide intermediate rounding, decimal variance, and scientific notation.

Extra functions such as conversions, random number generation, and standard deviations are available in pre-programmed applications packages. The full alphabet is included in the keyboard, which is used for making display-prompting programs and for convenient labeling of printout sections.

The basic calculator has 480 program steps and 40 data memories, and retails for $1695. The expanded memory option retails at $300.

For further information contact Texas Instruments, Inc., Post Office Box 5012, Dallas, TX 75222 (Attn: SR-60).

CIRCLE INQUIRY NO. 112

LITERATURE

Useful Catalogue Free

Abbott's new 1976-1977 Industrial Power Supply Catalog is now available, FREE.

This informative catalog has been expanded and contains complete physical and electrical specifications on our lines of AC to DC power supplies and DC to AC power inverters for industrial and OEM applications. All units are high reliability, rugged modules with built-in quality. Prices are listed with each model with discounts to 99 pieces shown.

For further information contact Abbott Transistor Laboratories, Inc., Industrial Products Division, 5200 West Jefferson Boulevard, Los Angeles, CA 90016; (213) 936-8185, Telex 69-1398.

CIRCLE INQUIRY NO. 113

Book Review

WHAT COMPUTERS CAN DO

By Donald D. Spencer

Camelot Publishing Co., Inc., 1977

Paperback

Review by
Judy Scolney Robertson &
Larry Robertson

What Computers Can Do is a general overview of the "practical applications of the computer and how it aids people in their everyday work." The book is aimed at a juvenile audience. It presupposes no familiarity with computers on any level and concentrates on a discussion of the various areas in which computers are used, saying very little about programming, design nor electronics.

Aside from the use of an occasional buzz word (e.g. "analog to digital," "transducer"), Spencer's approach is totally non-technical. Spencer covers numerous computer applications in this illustrated paperback. He includes education, banking, medicine, art, engineering, music, law enforcement, government, recreation and the media, as well as several other areas. In addition, Chapter 2, "The Computer Threat to Society," addresses such current problems as privacy and privileged information, the misuse of computers and the effects of automation on employment. This chapter is of particular interest to the young reader who will soon be emerging into a computer-dominated society.

Computers is a well-written introductory book for the preteen or teenager with no previous exposure to our friendly machines. However, the book could be improved by the inclusion of a definition of each technical term, either as it occurs or in an appended glossary. The illustrations, mostly photos, are interesting, but with the green on green printing, they are often poorly reproduced. Replacement of some photos with line drawings or cartoons would have significantly enhanced the book. This does not prevent us, however, from recommending ***What Computers Can Do*** as a fine answer to the question, "Why do we have to have computers?"

VOLTAGE REGULATOR HANDBOOK THEORY AND PRACTICE

Henry Wurzburg,
with contributing authors
Bernie Montoya, Cal Lidback
and Nick Lycoudes

Motorola Inc., 1976

Price: $2.50

Paperback. Available through Motorola distributors or by writing directly to Motorola Semiconductor Products Inc.

Review by
Judy Scolney Robertson &
Larry Robertson

The ***Voltage Regulator Handbook*** is a comprehensive guide to the use and functions of voltage regulators. Aimed at the electrical engineer, it is still comprehensible to the sophisticated hobbyist with a thorough knowledge of electronics. Written and published by Motorola Semiconductor Products Inc., the ***Handbook*** relies heavily on Motorola products in its detailed explanation of the operation of the voltage regulator. Motorola data sheets which are complete, clear and well designed are included in this highly technical manual, as is an industry-wide cross reference guide.

The need for voltage regulators is best stated in the preface, as well as a brief history and commentary on recent developments in the field:

> In many electronic systems, voltage regulation is required for certain functions. Yesterday's voltage regulators were often complex and expensive circuits. Valuable time was diverted from the major system development effort to design the voltage regulators. Today's monolithic integrated circuit (IC) regulator is easing that task. Available as a growing variety of special-function devices and with fixed and adjustable voltage ranges, the IC regulators offer design simplification and dramatic cost improvements.

Wurzburg, et al., more than adequately cover basic theory, selection of regulators, circuit configuration, design considerations, heat sinking, reliability, design and layout. In addition, special sections are devoted to trouble shooting and design of the input supply.

The ***Handbook*** is a totally professional publication, easy to refer to and to use. Additional sources are listed where appropriate, allowing the interested reader to find further data on the subject easily.

The ***Voltage Regulator Handbook*** is an exceptional reference for the use of voltage regulators. It would be a valuable, if not vital, addition to the technical library of anyone engaged in the selection and use of IC voltage regulators.

The Byte Shop reaches a new low in microcomputers.

$349

The Byt-8. It doesn't have a nifty (and expensive) front panel with lots of LED's and toggle switches.

And we obviously don't have a big full-color ad.

What we do have is the lowest priced microcomputer you can buy—built around the powerful and popular 8080A microprocessor.

For $349, you get the complete microcomputer card, motherboard, power supply and chassis in kit form.

The Byt-8 S100 bus is the same one used by Altair, IMSAI and most others so you have the greatest possible flexibility in choosing memory and input/output cards.

Optional cards from the Byte Shop include 4k, 8k or 16k of Random Access Memory, 4k or 8k of Programmable Read-Only Memory, a multiple input/output card, a TV typewriter card and, yes, a front panel bootstrap card, if you want the LED's and switches.

Even the CPU is optional. We'll sell you the chassis, motherboard and power supply for $229, and you can choose your own microcomputer card—a ZPU for instance?

Byt-8. It's the new low in price, but we're aiming for a new high in flexibility, delivery and support. See the Byt-8 at your nearest Byte Shop.

BYTE SHOP™
the affordable computer store

Stores now open in: **Arizona**, Phoenix, Tempe; **California**, Berkeley, Campbell, Fresno, Hayward, Lawndale, Long Beach, Mountain View, Palo Alto, Pasadena, Placentia, Sacramento, San Diego, San Francisco, San Mateo, Santa Barbara, Santa Clara, Tarzana, Thousand Oaks, Walnut Creek, Westminister; **Colorado**, Boulder, Englewood; **Florida**, Cocoa Beach, Miami; **Indiana**, Indianapolis; **Minnesota**, Eagan; **New York**, Levittown; **Oregon**, Beaverton, Portland; **Pennsylvania**, Bryn Mawr; **South Carolina**, Columbia; **Utah**, Salt Lake City; **Canada**, Vancouver, British Columbia. More Byte Shops are opening. Please write to Byte Incorporated, 1261 Birchwood Drive, Sunnyvale, California 94086.

CIRCLE INQUIRY NO. 41

THE BLACK

Imagine a black box. On one side is a switch, on the other a light. The switch has two positions, up and down. Place the switch up and the light comes on. Place it down and the light goes off.

Figure 1

Now try to imagine what is going on inside the box. First there might simply be two wires connecting the switch to the light. There is also the possibility that the switch does not connect directly with the light, but that it turns on a light inside to which a trained rat responds by running over and turning on another switch that turns on the outside light. A third possibility is that there is a human being inside the box who responds to an inside light in the same way the rat does.

These three possibilities demonstrate that there is more than one way of internally structuring a device to perform the same input-output function. A couple of important differences exist in these three forms.

One is speed. The wired switch turns on the light fastest in response to the switch being thrown. The rat is probably second in speed and the human is the slowest.

Figure 2

If the creator of the black box wanted to change the input-output function (e.g. to have the light go off when the switch is up and on when down), he would face different degrees of difficulty. In the wired case, he manually would have to change the connections between the switch and the light. In the rat example, he might either place a new rat in the box, trained in the opposite way, or quickly retrain the present one. For the human, he might simply say to him, "Okay, do it the other way," (being sure to specify exactly what is meant by "the other way").

These differences are for time and ease of change only; the input-output function remains the same. If one has ever used a computer system via a remote terminal or by passing cards and printout back and forth through a window, he cannot tell for sure whether the program went through a computer, whether a human figured out the answer and typed it back, or whether a highly trained rat responded to the input with a pre-learned response. In analogy, software and hardware and their in-between mixture, firmware, perform the same logical function. Their only difference is in speed and ease of change.

There is a second type of black box. This is one whose output is dependent not only upon the present input, but upon inputs which occurred in the past. For example, assuming the black box has the property that for every third time the switch is placed in the up position, the light is turned on, and it turned off the next time the switch came down. The sequence of events inside the box might be: "The switch is up, that's one. Now it's down. Now it's up, that's two. Now it's down. Now it's up again, turn on the light. Now it's down, turn the light off and start over."

Figure 3

This sequence may be viewed as a series of transformations in which the box's input-output function changes. Initially it is a box waiting for three "switch-ups" to occur in order to turn on the light. Once the switch has been thrown up, it is a box waiting for two more "switch-up's" before turning on the light. The second time the switch is thrown up, it is a box needing one more "switch-up." The third time the switch is put up, it is a box with the light on and requiring a single "switch-down" to turn it off.

Each of these forms represents the concept of a state. For each state, the response to present and

SW	Light
Up	On
Down	Off

Figure 1.

SW	Light
Up	Off
Down	On

Figure 2.

BOX

By KENNETH PUGH

future input differs. A state is dependent upon both the internal arrangements of the box and all previous inputs. A change in state may be considered as either a reconfiguration of the internal workings of the box or as some sort of memory (e.g. an "up-switch" counter).

There is a third type of black box which differs slightly from the previous ones. Suppose there was an additional input to the box which signaled to the internal operator that it must look at the input only when this additional input went from down to up or up to down. At any other time, the original input could be switched up and down and the state of the box and the light output would not be affected. This additional input signals the passage of time in the sense it tells the box the time at which to look at the other input. Thus it is commonly called a clock. Boxes whose state changes only when the clock is toggled (when signal goes from up to down or down to up) are termed synchronous. Most logic is built synchronously to avoid problems that occur when physical functions and interconnections are implemented.

Figure 4

One can explore the computer using the black box analogy in many ways, for it is nothing more than a complex interconnection of black boxes. Inputs may be represented by switches on a front panel, or a keyboard, or by electrical voltages or currents. Outputs may be lights or printed characters or electrical voltages or currents.

The designer of integrated circuits look at various silicon configurations as black boxes, providing input-output functions on a microscopic level. The logic designer uses the integrated circuits as black boxes to be interconnected to form a computer. The computer user may view the entire computer as a black box with a complex function of output versus input.

A computer user prepares a BASIC program to add two input numbers together and to print out the sum. He starts the computer and branches to a loader routine. By inputting the command "Load," he configures the computer into a black box that simply reads a binary input (the BASIC interpreter) and remembers it. By typing the input "Go," he changes the computer into a box that interprets his typed inputs as BASIC statements and remembers them. By typing "Run," he then changes his computer into a box that takes the next two inputs, adds them, and prints out the result.

The computer hobbyist may apply the black box approach on several levels; he can use a microprocessor circuit as a black box with its given input-output function and set of states defined by its internal registers, or he may buy a pre-built computer system and use it as a black box. He can buy a stored program in PROM (programmable read-only memory) and use it as a box to perform its function. Or he may use pre-written software (as a BASIC interpreter) as a black box.

One need not have any knowledge of an entire computer and a BASIC interpreter to use the BASIC language. By treating entire system as a set of black boxes, one can select those whose internal functions one wishes to understand and those which will be put to use.

The modern computer makes no distinction between instructions, which configure the device into a different black box, and data, which is input to the black box. Thus there is an ambiguity as to whether the computer responds to an input by changing itself into a different box or whether it simply remembers the past input. In either case, it can be considered a change in state.

Editor's note: . . . and from this series of microcosmic events, our lives in the everyday mesocosm have been irrevocably altered.

SW	# of 'UP'S'	Light
Down	0	Off
Up	1	Off
Down	1	Off
Up	2	Off
Down	2	Off
Up	3	On
Down	3	Off
Up	1	Off
	and so on	

Figure 3.

SW	Clock	Interval State
Up	↓	Previous
Up	↑	Up
Down	↓	Previous
Down	↑	Down

based on positive edge triggered service

Figure 4.

Software Section

By Robert A. Stevens
Software Editor

ALL MICROCOMPUTER MAGAZINES ARE NOT EQUAL

Contrary to Wayne Green's (Publisher of Kilobaud Magazine) generalized prognosis about microcomputer magazines not publishing long software programs nor paying more than $50.00 for software programs, other microcomputer magazines have been doing this for some time now. KILOBAUD's apparent policy of not publishing long programs nor paying more than $50.00 for a software program article in no way establishes the trend at the publishing market place.

BYTE, Dr. DOBB'S JOURNAL & INTERFACE AGE all have published long software programs in the past. Although I cannot speak for the other microcomputer magazines, INTERFACE AGE will continue to publish long programs. A good example of this is the 1976 December issue of INTERFACE AGE. This issue included the following software articles:

- Mark Borgerson's, Test Editor for the SWTPC-6800 software article consisting of 6½ typeset pages including a 605 line statement source listing program;
- Roger Rauskolb's Dr. Wang's Palo Tiny BASIC software article consisting of 10 typeset pages including grammar and over 1700 line statement source listing program;
- Four part serial LLL BASIC Interpreter software article that includes grammar and complete source listing program with over 200,000 bytes of source code that probably will require well over 50 typeset pages to publish.

Both of the other microcomputer magazines also have published meaningful long programs.

In regards to total payment for long software programs both BYTE and INTERFACE AGE are paying up to $50.00/typeset page for software manuscripts (Dr. Dobb's Journal current policy is not to pay for manuscripts published). Even a six year old would tell you that $50 times say 10 pages is more than $50 which is apparently the maximum price that KILOBAUD would pay for this software article.

To draw your own conclusions as to which microcomputer magazine to submit your software programs please refer to the following columns in this issue of INTERFACE AGE:

- INTERFACE AGE'S Best Article of the Month Award.
- INTERFACE AGE Will Pay Up to $50/Page for Software
- Best Article Of the Year Award

INTERFACE AGE IS FILLING THE SOFTWARE VOID WITH MAJOR PROGRAMS

INTERFACE AGE is filling that microcomputer software void. This issue includes six articles on software featuring a major stock investment program, and six software development programs. These programs are:

- The stock option program by Edward Christianson provides microcomputerized ***Hedge*** Options for the sophisticated stock investor. This software article includes fundamentals and presents stock investment strategies required to be successful at the market place. The software program is written in Processor Teck's 5K BASIC for the 8080 microcomputer
- A Random Number Generator assembly language program by Bob Martin provides an improved RND Function Generator for your 8080 BASIC Interpreter. In addition a RND Function Generator ***Chi-Square*** test program is included to help you determine just how good is your RND Function Generator.
- An 8080 Memory Object Code Search Routine in assembly language by T. E. Travis provides means for searching unknown object code for known instruction sequences in order to modify object coded software to be compatible with your system or to add software embellishments.
- A resident 6800 development software monitor program called PROTO developed by and made available to the readers of INTERFACE AGE by American Microsystems. PROTO is a ROM resident mon-itor firmware package that comes with AMI's EVK series of microcomputer boards.
- A BASIC Floating Point Math Package by David Mead and others for the LLL BASIC Interpreter is presented in this issue. This is part #3 of the series of articles on the LLL BASIC Interpreter program.

CALL FOR INFORMATION ON BASIC PROGRAMMING LANGUAGES

INTERFACE AGE is conducting a survey on the characteristics and programming power of microcomputer BASIC conversational programming languages. This survey includes Tiny BASIC (TB), Tiny BASIC Extended (TBX), Standard BASIC (SB), Standard BASIC Extended (SBX) and Business BASIC (BB) languages. One of the many objectives of this survey is to highlight the correlation between BASIC languages in order to provide insight for running a BASIC application program on any of the different BASIC languages. At the completion of this survey, the results will be published in INTERFACE AGE.

If you have developed, helped develop, or modified any BASIC type of programming language for any microcomputer, please contact or send hard copy of grammar, user's manual, copy of software and any supporting documentation to Robert A. Stevens, software editor, INTERFACE AGE. Please include your home and work telephone numbers (for coordination) with all correspondence.

BEST ARTICLE OF THE MONTH AWARD UP-DATE

INTERFACE AGE will bestow an Honorary Award of $100.00 to the author of the best non-commercial microcomputer article of the month. Only individuals are eligible for this monthly honorarium. This monthly

award is in addition to the honorarium given on the page count basis. Microcomputer articles may be on hardware, software or a combination hardware-software and will be judged by the INTERFACE AGE readership.

INTERFACE AGE NEEDS YOUR VOTE

Help INTERFACE AGE determine the type of articles you want to see published in the future by casting your vote of 10 points for the article (block voting) or articles (by vote splitting) you liked best. Feedback will provide encouragement to authors and will help make the INTERFACE AGE The Microcomputer magazine of the industry.

Each INTERFACE AGE magazine shall include one original bingo voting card. Each individual possessing a bingo voting card shall be allowed up to ten votes to be cast as a total single vote block for one author or subdivided into any vote block segment size combinations, with the total cast vote sum not to exceed ten, cast between two or more authors (no xerox copies of the bingo vote card please). Each published article is assigned a block of 10 bingo card numbers with the last digit of the number (LSD) to represent your cast vote value. 0 represents a vote value of 10. The prefix digits of the number block defines the article number.

All valid bingo vote cards must be postmarked prior to 12:00 P.M. of the last day of the month following the issue date of the related magazine.

BEST ARTICLE OF THE YEAR AWARD

INTERFACE AGE will bestow an Honorary Award of $500 value in products advertized in INTERFACE AGE to the author of the best non-commercial microcomputer article published during the year ending November 1977. The best article of the year award will be picked from the group of the best article of the month awards. Like the best article of the month award, only individuals are eligible for this yearly honorarium. The yearly award is in addition to the monthly award and honorarium given on the page count basis. The yearly award will be judged by the editors of INTERFACE AGE and will be announced in February 1978 issue of INTERFACE AGE.

INTERFACE AGE WILL PAY UP TO $50/PAGE FOR SOFTWARE

INTERFACE AGE is continually soliciting original unpublished quality documented highly commentated source/object code software listings and software technical articles for publishing in the INTERFACE AGE. Manuscript text must be typed double spaced with wide margins. Figures, tables, flow diagrams and charts must be numbered and submitted on separate sheets of white bond paper (Send original copy only). Program listings must be printed on white clean paper using a new black ink ribbon, and please, if possible, supply a punched paper tape assembly (source + object) code listing + source code listing + object code dump with your hard copy. Be sure to record your name, company and office and home telephone numbers on all materials submitted to the software editor. Also include statement in cover letter allowing INTERFACE AGE and the Microcomputer Software Depository to publish and distribute copies of your software program. Include a prepaid postage stamped envelope with your return address only if you want your manuscript returned, in the event that the submitted article is not accepted for publication.

Articles accepted and published will receive an honorary recognition award. Honorariums are based upon technical content, manuscript preparation and subject suitability for publication in INTERFACE AGE. Honoraria range from $15.00 to $50.00 per typeset magazine page. In addition, the best article of the month submitted will receive a $100 bonus. INTERFACE AGE'S readership will determine by vote which is the best article. (See best article of the month award). All software submitted to INTERFACE AGE will be deposited in the Microcomputer Software Depository (MDS) for low cost distribution.

Address all software correspondence to R. A. Stevens, Software Editor, c/o INTERFACE AGE Magazine, 2361 E. Foothill Blvd., Pasadena, CA 91107 or call (213) 449-1655.

SOFTWARE SHOPPING LIST

Now that INTERFACE AGE has expanded the microcomputer software coverage and developed a large appetite for good software, your programs and application software is badly needed to sate this enlarged software appetite. This software shopping list includes the following:

- **Microcomputer Development Software** such as assemblers, disassemblers, editors, monitors, utilities, mini-maxi BASIC interpreters and compilers, FORTRAN interpreters and compilers, boot strap loaders, software drivers, cassette software operating systems (COS), floppy disc software operating systems (FDOS), TTY software operating systems (TTYOS), and CRT software operating systems (CRTOS) for all microcomputer configurations.
- **Short Software Routines** such as math packages and I/O diagnostics for all microcomputer configurations.
- **Application Software Programs** such as Analog To Digital Converter (DAC) - Digital to Analog Converter (DAC) software control programs, Automated membership billing and mailing list update program, Inventory control software, invoice and billing software. Accounts Receivables and Payable software, process control programs, etc. for all microcomputer configurations.
- **Software Communications Protocol Programs** for such communication protocols as the BSC or Bisync (Binary Synchronous communications protocol procedures) and the new SDLC (Synchronous Data Link Control communication protocol procedures), etc.
- **Off-line Software Storage Format Control Programs** for cassette, paper tape, and floppy disc software record-

ing formats.

- **Microcomputer Game Programs**
- **Add Your Own Program Shopping List Here** Send your list to the software editor.

HALF SIZE PROGRAM LISTINGS VS FEWER PROGRAMS UPDATE

Starting with this issue full size xerox copies of software published in half size format will be available from the Microcomputer Software Depository (MSD). See MSD program listing for details.

INEXPENSIVE MICROCOMPUTER SOFTWARE

The Microcomputer Software Depository (MSD) will act as repository for source and object code tapes. Programmers wishing to contribute programs to the public domain but who do not want to bother with distribution, may do so by forwarding appropriate documentation including short descriptive write-up and punch paper tape copy of program if possible or cassette copy to MSD. There is no membership fee for access to the public domain paper tapes (PDT) from MSD.

Anyone may obtain copies of these PDT software packages by prepaying a small fee with the order to cover duplication, postage and handling cost. Prices will be listed periodically in INTERFACE AGE. Typical cost for a short program will be approximately $2.00 ($2.00/ounce) + tax, postage and handling. As a convenience MSD will also provide punched paper tape copies of vendor supplied software packages (VSP) that will be sold at vendor suggested sale prices. For a current copy of the available software from the Microcomputer Software Depository (MSD) send a check for $1.00 with a prestamped return envelope to MSD.

Support MSD to build a software library by sending copies of your documented software programs including short description, flow diagrams and punched paper tape source code and object listings if possible or cassette tape copy for low cost distribution to the following address: Microcomputer Software Depository, 2361 E. Foothill Blvd., Pasadena, CA. 91107; (213) 449-0616.

The following lists software available from MSD on a prepaid basis only. The total cost of each package is the sum of the basic price + California sales tax, if applicable, + postage and handling cost. Foreign subscribers please note the different mailing cost for postage outside USA. Address all inquiries to

Microcomputer Software Depository
2361 E. Foothill Blvd.
Pasadena, Calif., 91107
or call (213) 449-0616

MICROCOMPUTER SOFTWARE DEPOSITORY (MSD) PROGRAMS DATE FEB. 1977 REV. 1

PROGRAM MEDIA

PTAC	PAPER TAPE ASSEMBLY CODE
PTSC	PAPER TAPE SOURCE CODE
PTOC	PAPER TAPE OBJECT CODE
PTBC	PAPER TAPE BASIC CODE
PTAL	PAPER TAPE ASSEMBLY LISTING
PTSL	PAPER TAPE SOURCE LISTING
PTOL	PAPER TAPE OBJECT LISTING
PTOD	PAPER TAPE OBJECT DUMP
PTBL	PAPER TAPE BASIC LISTING
CTAL	CASSETTE TAPE ASSEMBLY LISTING
CTSL	CASSETTE TAPE SOURCE LISTING
CTOL	CASSETTE TAPE OBJECT LISTING
CTOD	CASSETTE TAPE OBJECT DUMP
CTBC	CASSETTE TAPE BASIC CODE
CTBL	CASSETTE TAPE BASIC LISTING
HCAC	XEROX HARD COPY OF ASSEMBLY CODE
HCSC	XEROX HARD COPY OF SOURCE CODE
HCOC	XEROX HARD COPY OF OBJECT CODE
HCBC	XEROX HARD COPY OF BASIC CODE
HCAL	XEROX HARD COPY OF ASSEMBLY LISTING
HCALF	FULL SIZE XEROX HARD COPY OF ASSEMBLY LISTING
HCSL	XEROX HARD COPY OF SOURCE LISTING
HCOL	XEROX HARD COPY OF OBJECT LISTING
HCOD	XEROX HARD COPY OF OBJECT DUMP
HCBL	XEROX HARD COPY OF BASIC LISTING
TEXT	XEROX HARD COPY OF PRINTED TEXT
PTTL	PAPER TAPE TEXT LISTING
CTTL	CASSETTE TAPE TEXT LISTING
MAN	MANUAL
HCGR	XEROX HARD COPY OF GRAMMAR
PTGR	PAPER TAPE COPY OF GRAMMAR
BBSL	XEROX HARD COPY OF BINARY BOOTSTRAP LOADER
HBSL	XEROX HARD COPY OF HEX BOOTSTRAP LOADER
PACK	PACKAGE PRICE INCLUDES ALL ITEMS/PROGRAM # WITH SYMBOL <

SUFFIX C = HAND ASSEMBLED CODE
SUFFIX L = COMPUTER FORMATED LISTING
SUFFIX D= CODE DUMP IN OCTAL OR HEX
SUFFIX F= FULL SIZE COPY

NOTES

*	CALIF. SALES TAX REQUIRED FROM RESIDENTS OF CALIF. USA POSTAGE + HANDLING OR THIRD CLASS USA POSTAGE + HANDLING OR SURFACE RATE FOREIGN POSTAGE @ THREE TIMES THIRD CLASS USA POSTAGE RATE (STANDARD) OR SURFACE RATE FOREIGN POSTAGE @ FIVE TIMES USA POSTAGE RATE (ALTERNATE)
>	NEW PROGRAM LISTING
%	VENDOR SOFTWARE PACKAGE

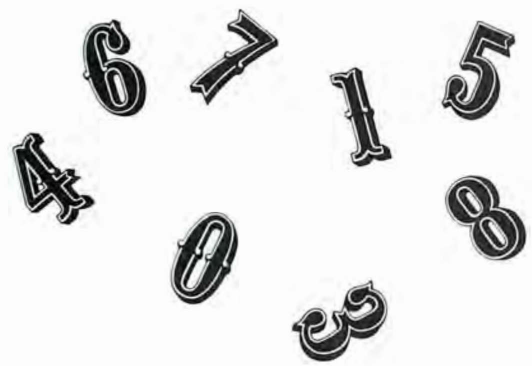
DEFINITIONS:

ASSEMBLY LISTING: COMPUTER ASSEMBLED SOFTWARE PROGRAM LISTING THAT INCLUDES SYMBOLIC ASSEMBLY LANGUAGE SOURCE CODED INSTRUCTIONS WITH COMMENTS PLUS EQUIVALENT MACHINE LANGUAGE OBJECT CODED INSTRUCTIONS AND MEMORY ADDRESS ASSIGNMENTS FOR EACH INSTRUCTION (SOURCE + OBJECT).

ASSEMBLY CODE: SAME CONTENT AS ASSEMBLY LISTING BUT HAND ASSEMBLED.

SOURCE LISTING: SOFTWARE PROGRAM LISTING RESULTING FROM COMPUTER SOFTWARE CONTROLLED ASSEMBLY PROCESS THAT INCLUDES ASSEMBLY LANGUAGE SOURCE CODED INSTRUCTIONS WITH COMMENTS. SOMETIMES, LINE STATEMENT NUMBERS ARE INCLUDED FOR EACH INSTRUCTION.

BRANCH TO PAGE 101



Have you ever been suspicious of the numbers that your version of BASIC pumps out for use in your favorite computer game? Maybe the royal rats are consistently eating too much grain in the game of KINGDOM, or possibly you think your shields are absorbing energy at an unreasonable rate in STAR TREK. If this sounds familiar, I suggest you pay a visit to your favorite psychiatrist and get checked out for a persecution complex. The only way you could guess that your random number generator was a lemon is if it were written by Maxwell's Demon, the infamous violater of the Second Law of thermodynamics.

Need A Better Random Number Generator?

By Bob Martin

There is a right number, a wrong number and a random number. But when is the right number a wrong number for a random number?

Having said that, I am going to claim anyway that I did detect a bad one in this manner. My guess that it was a bad generator, turned out to be correct. A few weeks back I wrote a Yahtzee game in order that my wife could play with the computer. (This is a good practice. It helps to reassure her that the money we spent on it was well worth it.) While checking out the game for errors I noticed, more often than I could believe, that the game ended with the sixes box empty. How many games with zero in the sixes box would be believable?

The RND function in BASIC returns a decimal fraction that lies between zero and one. Now suppose you have ten boxes and four red marbles . . . No wait! I'm not going to do that. Everybody falls asleep when the boxes and marbles are brought out. Let's assume that everyone reading this likes programming. Therefore we have an array of ten elements numbered zero through nine. All ten elements start out with the value zero. We will now call upon the RND function to decide which of the ten elements to increment by one. This is very simple. Just let X = INT (10*RND(0)). X will now be the number of the element that is to be incremented. Now do this for, say, 1000 calls of RND. If RND generated truly random numbers, would it be believable if each element had the value 100? A person who had not studied elementary statistics might be surprised to learn that this would be very unbelievable.

To describe just how unlikely it is that the above result was produced by a truly random generator, let me change the outcome slightly. I will change it to a result that is much more likely to have been produced by a real random number generator. Suppose the ten elements turned out to have the following values: four of them were 95, four were 105, one was 99 and one was 101. A truly random RND function would give a result, this close to the expected result, only once in 100 trials of the experiment. (Each experiment, remember, was 1000 calls of RND.) Yet this modified result is very much more probable than the case where all ten elements have the value 100.

RND FUNCTION *CHI-SQUARE* TEST

You might wonder how such things can be deter-

mined. It is made possible by using one of the oldest statistical tests there is. This test is called the ***chi-square*** test and was introduced by Karl Pearson in 1900. We don't care right now why it works. Let that be a question to motivate the reading of some mathematics. For the present we only want to know if our random number generator can pass the ***chi-square*** test.

To make this test we must calculate a number called the ***chi-square statistic***. This calculation is very simple for a random number generator. First, pick the number of categories you wish to use. In the above example the number was ten. Next choose the number of independent observations to be made. We used 1000. Now, since there are 1000 numbers that can fall into 10 categories, we would expect there to be 100 in each of them. We will call this the ***expected*** value even though we know better than to expect it too often. Take the ***expected*** value and subtract it from the value observed in each category. The observed values in the example above are the values that the elements of the array had after 1000 calls of RND. Take the result of each of these subtractions and square it. Now divide each of them by the expected value. To get the ***chi-square statistic***, add all these numbers together. In the example above, you will have to sum ten numbers, one for each category. The ***chi-square statistic*** for the modified example above is 2.02.

HOW TO READ THE TEST RESULTS

Now you want to know what this number tells you about your random number generator. That's easy. It has all been worked out for you in a table. If you do not like tables, fear not. The ***chi-square*** distribution table is simple. The only thing you need to know is that (for our problem) the number of degrees of freedom is one less than the number of categories. We had ten categories, therefore we had nine degrees of freedom. Find the row in the table labeled with the number 9. This is the only row of data that will pertain to your generator. (You would use a different row if you had chosen a number of categories other than ten.) Now find the place your ***chi-square statistic*** falls in this row of data. Each number in the row also has a number at the top of its column. This is the number that interests you. It will be either a percentage figure or a decimal fraction less than one.

To illustrate, suppose your ***chi-square statistic*** is 2.09. Look for the number 2.09 in the row labeled 9. At the top of the column that contains 2.09 you will find either .99 or 99%. What this number means is the following: If your random number generator was truly random, then 99% of the time it would produce a number greater than 2.09. It is now up to you to decide if this is sufficiently random for your intended use of the generator. Since 99% is most of the time, you probably would not accept this. The same goes for the other end of the row. If you came up with the figure 1%, it would mean the ideal generator would get a ***chi-square*** larger than yours only 1% of the time.

A simple way to proceed is to declare the generator is acceptable if it falls within the 95 to 5 percent range. For the 9 degrees of freedom we have been using, this amounts to saying we will accept the generator if it can produce a ***chi-square statistic*** that lies between 3.3 and 16.9. The BASIC program given below may be used to test your random number generator. It uses 10,000 trials of RND and puts them in 10 categories. Notice that the number of trials has no effect on how the table is used. Only the number of degrees of freedom is important.

I tried this program with my three versions of BASIC. The results were so poor (***chi-square*** of 217 was typical) that I had to recheck the ***chi-square*** program for errors. A moment's thought, of course, will give the reason why they were all so poor. Why take up more space than you have to in a BASIC interpreter, if most people only use RND to play STAR TREK? I can think of a couple of possible reasons; one is that someday you might want to use RND for something besides playing games. Another is that a bad RND might cause you to lose more games than is good for your self-image. Even if you do always win easily, the games will not be fun for long when you know that the reason you win is a faulty RND.

JUST HOW GOOD IS YOUR RND FUNCTION GENERATOR?

The bottom line is that it is simply more satisfying to have a good RND. Yahtzee feels better if you know your computer dice are as good as real ones. So try the test program (unless you wish to remain forever in doubt) and if your RND fails, I have a solution below for you. I might give one word of caution here that will serve as an incentive to read on. Even if your RND passes the test, it is not insured that your generator is perfect. There is a whole battery of tests that may be run. One such test is called the serial test which checks to make sure you don't have too many ascending or descending sequences of numbers. A socalled ***feedback*** generator will often fail this test. The generator I give below in 8080 code can pass many of the tests for randomness.

A BETTER RND FUNCTION GENERATOR

The method used here, to generate a sequence of random numbers, is a very common one called the mixed linear congruential method. It works like this. The previous random number (you must supply the very first number called the ***seed***) is multipled by a number called the ***multiplier***. To this is added another number called the ***constant***.. Then still another number called the ***modulus*** is repeatedly subtracted from the result until the result is less than the ***modulus.*** This will then be the new random number. The whole process is repeated over and over, producing a sequence of numbers all of which are less than the ***modulus***. If each new random number is always divided by the ***modulus***, the result will be a sequence of numbers that are just what RND is supposed to produce, a sequence of random numbers that all lie between zero and one. It will be a random sequence provided that ***multiplier, constant*** and ***modulus*** are chosen properly.

A look at the 8080 RND function generator program will reveal two familiar strings of digits used for the ***multiplier*** and ***constant***. The number used for the ***modulus*** is 2 raised to the 35th power. This makes it especially easy to do the repeated subtraction of the ***modulus*** when the arithmetic is done in binary. In fact, it

There are many areas to consider for the implementation of a good random number generator

```
2900 CDBB29  RND:   CALL ZERO      ;CLEAR WORK SPACE
2903 21A529         LXI H,MULT+3   ;POINT TO MULTIPLIER
2906 11AF29         LXI D,RSPC+4   ;POINT TO LAST RND NUM
2909 0605           MVI B,5        ;SET BYTE COUNT
290B 0E08           MVI C,8        ;SET BIT COUNT
290D C5      BITC:  PUSH B         ;SAVE BYTE COUNT
290E D5      SWD:   PUSH D         ;SAVE BEG OF ADD
290F E5             PUSH H         ;CURRENT BYTE OF MULTIPL
2910 7E             MOV A,M        ;GET BYTE OF MULTIPLIER
2911 0F             RRC            ;ROTATE IT
2912 77             MOV M,A        ;PUT IT BACK
2913 D22929         JNC NOBT       ;JUMP IF NOTHING TO A
2916 2156D6         LXI H,-RSPC-1  ;LIMIT OF RND NUM
2919 19             DAD D          ;CALC. COUNT TIL END
291A 45             MOV B,L        ;SAVE COUNT
291B 210500         LXI H,5        ;RESULT GOES -
291E 19             DAD D          ; 5 BYTES DOWN
291F 1A      ADCL:  LDAX D         ;GET NEXT BYTE TO
2920 8E             ADC M          ;ADD TO RESULT
2921 77             MOV M,A        ;SAVE IT
2922 2B             DCX H          ;NEXT BYTE OF RE
2923 1B             DCX D          ;NEXT BYTE TO A
2924 05             DCR B          ;ONE BYTE DOWN
2925 C21F29         JNZ ADCL       ;LOOP IF NOT T
2928 AF             XRA A          ;CLEAR CARRY
2929 11AF29  NOBT:  LXI D,RSPC+4   ;POINT TO ST
292C 0605           MVI B,5        ;SET COUNT
292E 1A      SHFR:  LDAX D         ;GET NEXT B
292F 17             RAL            ;SHIFT LEF
2930 12             STAX D         ;PUT IT BA
2931 1B             DCX D          ;ON TO NE
2932 05             DCR B          ;DCR COU
2933 C22E29         JNZ SHFR       ;LOOP IF
2936 E1             POP H          ;RESET
2937 D1             POP D          ;RESET
2938 0D             DCR C          ;ONE P
2939 C20E29         JNZ SWD        ;LOOP
293C C1             POP B          ;GET
293D 2B             DCX H          ;ANC
293E 1B             DCX D          ;AD
293F 05             DCR B          ;DC
2940 C20B29         JNZ BITC       ;I
                    ;
             ;ADD THE CONSTANT TO THE
                    ;
2943 01AA29         LXI B,CONS+3
2946 21B429         LXI H,WKSP+4
2949 11AF29         LXI D,RSPC+4
294C AF             XRA A
294D 3E05           MVI A,5
294F 32BA29         STA CNT
2952 0A      ACON:  LDAX B
2953 8E             ADC M
2954 12             STAX D
2955 0B             DCX B
2956 1B             DCX D
2957 2B             DCX H
2958 3ABA29         LDA CNT
295B 3D             DCR A
295C C24F29         JNZ AC
                    ;
             ;CONVERT BIN
                    ;
295F CDBB29         CAL
2962 11AB29         LX
2965 1A             LI
2966 OF             R
2967 OF
```

doesn't need to be done at all. Just ignore all bits higher than 35 and the ***modulus*** operation is taken care of automatically. Since the number of bits is finite we cannot produce an infinite sequence of numbers. At some point the generator must repeat a number that was produced earlier. When this happens the whole pattern will also be repeated because the current number is always constructed from the immediately preceding number.

The above generator will give through all possible 35 bit numbers before it repeats. You may have noticed that the ***familiar*** numbers used are not what they should be in the last digit of each. They were changed so as to meet the conditions of a theorem that guarantees the generator will go without repeating as long as possible. In this case it will run for about 2 to the 35th or 34 billion numbers. To call the RND function this many times on my little micro would consume 11 years of around-the-clock computer time. Of course a long period is no guarantee of randomness. Counting from zero to 34 billion gives a sequence of long period which is obviously not random. The parameters given above provide for both a long period and randomness.

APPLICATION OF RND FUNCTION GENERATOR

There are a couple of things you must know to use this generator. If you want to repeat a given sequence of numbers, then locations 29AB through 29AF must be initialized with the same number each time you want to run the sequence. The fact that a sequence is reproducible is the reason these generators are called ***psuedo-random***. If you don't care about being able to repeat a sequence then don't worry about what ***seed*** is at these locations. Since the generator is of maximum period there is no ***seed*** than can get you hung up in a shorter period sequence.

Another thing to know concerns returning the random number to your BASIC interpreter. The last part of the routine converts the binary random number in 29AB - 29AF into a BCD number and puts it in locations 29B0 through 29B5. This number must now be made compatible with the floating point representation of your BASIC interpreter. This is necessary because the BCD number must be divided by the ***modulus*** so that the final result lies between 0 and 1. A better method is to multiply by the inverse of the ***modulus*** (2 to the minus 35th or 2.91038E-11). Jump to your routine that does this preparation and multiplication from location 299F. A return after the multiplication should return to whatever called RND.

Remember that if your BASIC uses only six significant digits then only six digits of the BCD number are required. This will mean that a given number returned by RND will show up more often than once every 34 billion calls. However the pattern in the sequence before and after it will be different.

RND FUNCTION APPLICATIONS

Now that you have a fairly decent RND, what can you do with it besides play games? One demonstration is a simple example of what is called a Monte Carlo simulation. Try to calculate the number Pi by using RND: Draw a unit circle. Enclose the circle with a

square that touches it at four points. Now use RND, in a BASIC program, to shoot darts over the entire area of the square. Count both the total number of darts thrown and the number of these that fall inside the circle. The ratio of the darts inside to the total number is the same as the ratio of the area of the circle to that of the square. The area of the circle is just Pi and the area of the square is four. Therefore multiply the dart ratio by four to get Pi.

If you should want to try this, here are two things to make it easier. First, use only the first quadrant. Second, when deciding if the dart is inside or outside the circle, there is no need to take the square root (unit circle). When I tried this on my computer with 16,000 darts 14 different times, I got the following results: Old bad RND, Pi=3.07778; New good RND, Pi=3.14128. Both had a standard deviation of .01. This method may be used to find volumes, masses, centers of mass etc., on more complcated objects that cannot be calculated exactly by other means.

Here is one more application which, even though unrealistic, might give you some other ideas. Suppose you know the average lifetime and the standard deviation of the chips in your 65K memory system. As the time approaches for the end of the first lifetime, more and more chips must be replaced one at a time. You might wonder if there is a time when it would be cheaper to just replace all the memory at once. Factors that would influence you to make a total replacement are: (1) discount price on large quantity of chips; (2) no time is wasted hunting for bad chips. Of course this is unrealistic because these factors would have to be greatly exaggerated before even one total replacement would be considered reasonable.

One way you might want to simulate this problem is to consider each chip separately. As you come to each chip you want an answer to the question, "How long before this chip will break down?" To get the answer just use the BASIC subroutine given below. Pass it the mean lifetime of a chip (M1) and the standard deviation of this lifetime (D1) and it will return the amount of time until the next breakdown (X) for that chip. The subroutine uses the Center Limit approach to simulate a normal distribution with the given parameters. Now run through the history of each chip for a given period of time, say, five years. Record the cost of each individual breakdown and the cost of each replacement of all 65K. The number of times the 65K is replaced in five years is determined by the time between replacements that you have selected. Run through all the chips for the five year simulation with your first selection for this amount of time. Then try it again for another time; and another. Vary this time (in a systematic fashion) until the toal cost of maintenance for a five year period is a minimum. When the minimum is found you have the answer to your problem.

SUPER RND FUNCTION GENERATOR

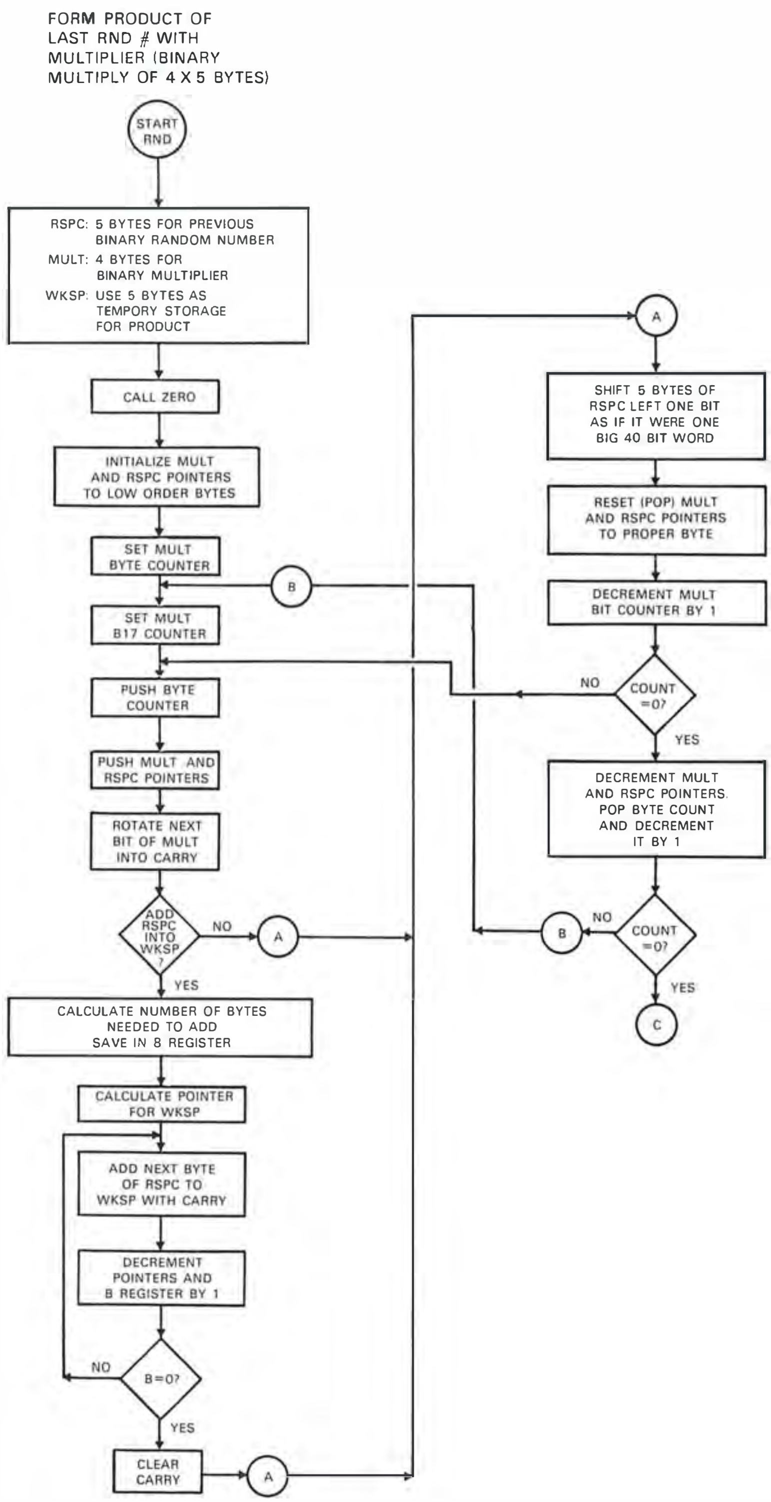
One final note. If by chance the given generator should fail to be random in some application you might dream up, there is a plan for the Super Random

Number Generator. It was suggested by McLaren and Marsaglia and is said to meet anyone's criteria for ***psuedo-randomness***. To accomplish this, two generators are required. Let the generator given above be called generator A. To construct generator B, simply exchange the values of ***multiplier*** and ***constant*** of generator A. Use generator A to lay out a list of about 100 random numbers. Use generator B to decide which of these 100 will be the next in our Super Sequence. The number which is used will be removed from the list and replaced by a new number given by generator A. The whole process is repeated for the next number in the Super Sequence.

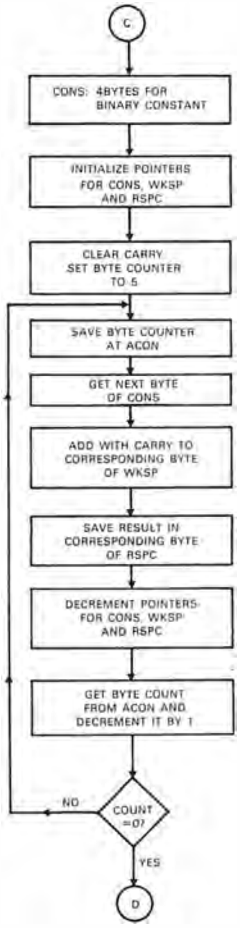
In summary I wish to encourage you to dump that old mantra you have been using and begin chanting RND to experience the true chaos of the universe.

SEE MICROCOMPUTER SOFTWARE DEPOSITORY PROGRAM INDEX FOR COPIES OF THIS PROGRAM

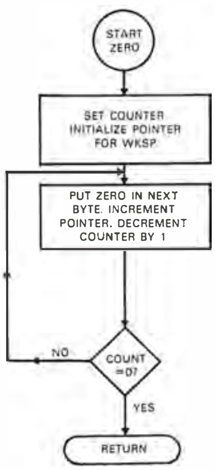
RND FUNCTION GENERATOR FLOW DIAGRAMS



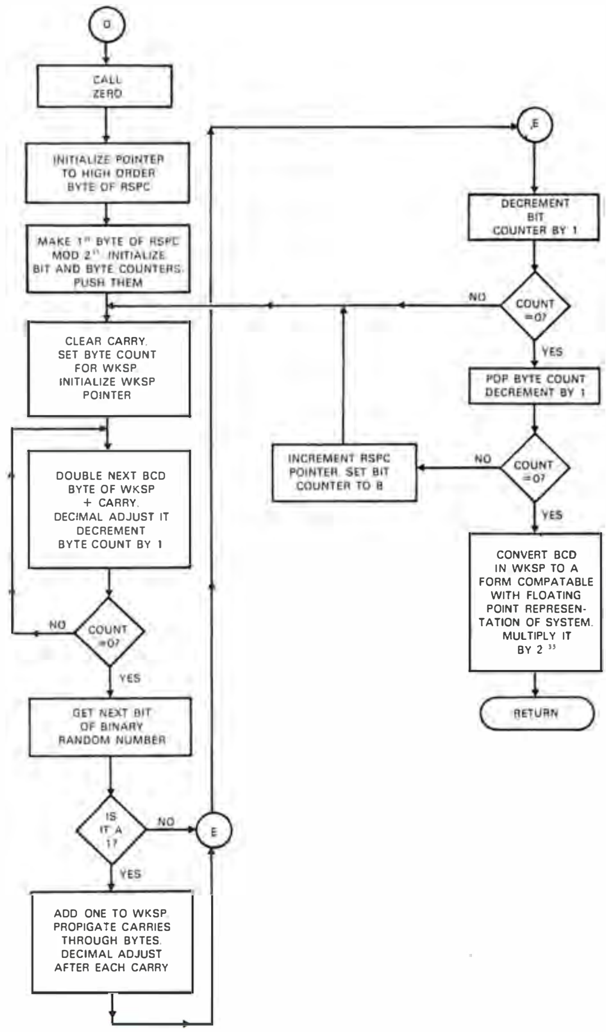
ADD THE CONSTANT TO
THE RESULT (WKSP)
BINARY ADD OF 4 BYTES



FILL WKSP
WITH ZEROES



CONVERT BINARY RANDOM
NUMBER TO BCD BY
POLYNOMIAL EVALVATION
USING HORNER'S METHOD



Add the constant to the result (WKSP) binary ADD of 4 bytes

Convert binary random number to BCD by polynominal evaluation using Horner's method

RND FUNCTION GENERATOR PROGRAM

```
                TITLE  /RANDOM NUMBER GENERATOR/
               ;
               ;THIS ROUTINE GENERATES RANDOM NUMBERS BY
               ;THE LINEAR CONGRUENTIAL METHOD (MIXED)
               ;
               ;FORM THE PRODUCT OF THE LAST RANDOM
               ;NUMBER WITH THE MULTIPLIER
               ;
2900 CDBB29    RND:    CALL ZERO         ;CLEAR WORK SPACE
2903 21A529            LXI H,MULT+3      ;POINT TO MULTIPLIER
2906 11AF29            LXI D,RSPC+4      ;POINT TO LAST RND NUM
2909 0605              MVI B,5           ;SET BYTE COUNT
290B 0E08      BITC:   MVI C,8           ;SET BIT COUNT
290D C5                PUSH B            ;SAVE BYTE COUNT
290E D5        SWD:    PUSH D            ;SAVE BEG OF ADD
290F E5                PUSH H            ;CURRENT BYTE OF MULTIPLIER
2910 7E                MOV A,M           ;GET BYTE OF MULTIPLIER
2911 0F                RRC               ;ROTATE IT
2912 77                MOV M,A           ;PUT IT BACK
2913 D22929            JNC NOBT          ;JUMP IF NOTHING TO ADD
2916 2156D6            LXI H,-RSPC-1     ;LIMIT OF RND NUM
2919 19                DAD D             ;CALC. COUNT TIL END
291A 45                MOV B,L           ;SAVE COUNT
291B 210500            LXI H,5           ;RESULT GOES -
291E 19                DAD D             ; 5 BYTES DOWN
291F 1A        ADCL:   LDAX D            ;GET NEXT BYTE TO ADD
2920 8E                ADC M             ;ADD TO RESULT
2921 77                MOV M,A           ;SAVE IT
2922 2B                DCX H             ;NEXT BYTE OF RESULT
2923 1B                DCX D             ;NEXT BYTE TO ADD
2924 05                DCR B             ;ONE BYTE DOWN
2925 C21F29            JNZ ADCL          ;LOOP IF NOT DONE
2928 AF                XRA A             ;CLEAR CARRY
2929 11AF29    NOBT:   LXI D,RSPC+4      ;POINT TO START AGAIN
292C 0605              MVI B,5           ;SET COUNT
292E 1A        SHFR:   LDAX D            ;GET NEXT BYTE OF LAST RND NUM
292F 17                RAL               ;SHIFT LEFT
2930 12                STAX D            ;PUT IT BACK
2931 1B                DCX D             ;ON TO NEXT BYTE
2932 05                DCR B             ;DCR COUNT
2933 C22E29            JNZ SHFR          ;LOOP IF MORE
2936 E1                POP H             ;RESET H TO MULTIPLIER
2937 D1                POP D             ;RESET D FOR ADD
2938 0D                DCR C             ;ONE BIT OF MULT DONE
2939 C20E29            JNZ SWD           ;LOOP IF NOT DONE
293C C1                POP B             ;GET BYTE COUNT
293D 2B                DCX H             ;ANOTHER BYTE DONE
293E 1B                DCX D             ;ADD POINT MOVES DOWN 1 BYTE
293F 05                DCR B             ;DCR COUNT
2940 C20B29            JNZ BITC          ;LOOP IF MORE BYTES OF MULT
                       ;
               ;ADD THE CONSTANT TO THE RESULT
               ;
2943 01AA29            LXI B,CONS+3      ;POINT TO CONSTANT
2946 21B429            LXI H,WKSP+4      ;RESULT OF MULT
2949 11AF29            LXI D,RSPC+4      ;PLACE FOR NEW RND NUM
294C AF                XRA A             ;CLEAR CARRY
294D 3E05              MVI A,5           ;SET COUNT
294F 32BA29    ACON:   STA CNT           ;SAVE IT
2952 0A                LDAX B            ;GET NEXT BYTE
2953 8E                ADC M             ;ADD THE PRODUCT
2954 12                STAX D            ;SAVE RESULT
2955 0B                DCX B             ;BUMP
2956 1B                DCX D             ;ALL
2957 2B                DCX H             ;POINTERS
2958 3ABA29            LDA CNT           ;GET COUNT
295B 3D                DCR A             ;DECREMENT IT
295C C24F29            JNZ ACON          ;LOOP IF ANY LEFT
               ;
               ;CONVERT BINARY RANDOM NUM TO BCD
               ;
295F CDBB29            CALL ZERO         ;CLEAR WORK SPACE
2962 11AB29            LXI D,RSPC        ;POINT TO BIN RND NUM
2965 1A                LDAX D            ;GET FIRST BYTE
2966 0F                RRC               ;MAKE IT -
2967 0F                RRC               ;MOD -
2968 0F                RRC               ;2**35 (POWER)
2969 12                STAX D            ;PUT IT BACK
296A 010305            LXI B,503H        ;PREPARE TO JUMP -
296D C5                PUSH B            ;INTO THE MIDDLE
296E C37529            JMP DLB1          ;DO IT
2971 13        DLB:    INX D             ;NEXT BYTE OF BIN RND
2972 C5                PUSH B            ;SAVE BYTE COUNT
2973 0E08              MVI C,8           ;BIT COUNT
2975 AF        DLB1:   XRA A             ;CLEAR CARRY
2976 0606              MVI B,6           ;BYTE COUNT FOR WKSP
2978 21B529            LXI H,WKSP+5      ;LOW ORD OF BCD RND NUM
297B 7E        TTIM:   MOV A,M           ;GET NEXT 2 DIGITS
297C 8F                ADC A             ;DOUBLE THEM
297D 27                DAA               ;MAKE IT DECIMAL
297E 77                MOV M,A           ;PUT IT BACK
297F 2B                DCX H             ;MOVE TO NEXT HIGHER ORDER
2980 05                DCR B             ;DCR COUNTER
2981 C27B29            JNZ TTIM          ;LOOP IF MORE
2984 1A                LDAX D            ;GET BYTE OF BIN RND NUM
2985 07                RLC               ;CHECK FOR BIT
2986 12                STAX D            ;PUT IT BACK
2987 D29629            JNC MBT           ;JUMP IF NO BIT
298A 21B529            LXI H,WKSP+5      ;BCD RND NUM
298D 7E        PLUS:   MOV A,M           ;GET BYTE OF RND NUM
298E CE00              ACI 0             ;PROPIGATE THE CARRY
2990 27                DAA               ;DECIMALIZE
2991 77                MOV M,A           ;PUT IT BACK
2992 2B                DCX H             ;ONE DOWN
2993 DA8D29            JC PLUS           ;DONE IF NOTHING TO CARRY
2996 0D        MBT:    DCR C             ;DONE WITH 1 BIT OF BIN NUM
2997 C27529            JNZ DBL1          ;LOOP IF MORE IN THIS BYTE
299A C1                POP B             ;GET BACK BYTE COUNT
299B 05                DCR B             ;DECREMENT IT
299C C27129            JNZ DLB           ;LOOP IF ANY LEFT
299F C3XXXX            JMP               ;GOTO DIV ROUT.- U SUPPLY
29A2 BB40      MULT:   DW 403BH          ;3141592653
29A4 E64D              DW 4DE6H          ;THE MULTIPLIER
29A6 00                DB 0              ;ZERO NEEDED HERE
29A7 A205      CONS:   DW 5A2H           ;2718281829
29A9 B065              DW 65B0H          ;THE CONSTANT
29AB           SPC:    DS 1              ;STORAGE FOR BIN RND NUM
29B0           WKSP:   DS1               ;STORAGE FOR BCD RND NUM
29BA           CNT:    DS1               ;XTRA COUNT
               ;
               ;ROUTINE TO CLEAR WKSP
               ;
29BB 01000A    ZERO:   LXI B,0A00H       ;COUNTER AND ZERO
29BE 21B029            LXI H,WKSP        ;PLACE TO BE CLEARED
29C1 71        ZER1:   MOV M,C           ;PUT ZERO IN NEXT BYTE
29C2 23                INX H             ;NEXT BYTE
29C3 05                DCR B             ;DOWN WITH COUNT
29C4 C2C129            JNZ ZER1          ;LOOP TIL DONE
29C7 C9                RET               ;BACK TO WORK
```

RND FUNCTION GENERATOR *CHI-SQUARE* TEST PROGRAM

```
5 REM CHI-SQUARE TEST
10 DIM A(9)
20 K=10
30 N=10000
35 REM ZERO THE ELEMENTS
40 FOR J=0 TO K-1
50 A(J)=0;NEXT J
55 REM LET RND FILL THE ARRAY
60 FOR J=1 TO N
70 B=INT(K*RND(0))
80 A(B)=A(B)+1
90 NEXT J
95 REM CALC. CHI-SQUARE
100 D=0
110 FOR J=0 TO K-1
120 C=A(J)-N/K
130 D=D+C*C
140 NEXT J
150 PRINT "CHI-SQUARE STATISTIC = ",K*D/N
160 STOP

500 REM THIS ROUTINE RETURNS A
501 REM NORMALY DISTRIBUTED RANDOM
502 REM VARIATE WITH PARAMETERS D1 AND M1
505 X=-6
510 FOR K=1 TO 12
520 X=X+RND(0)
530 NEXT K
540 X=X*D1+M1
550 RETURN
```

VECTORED FROM PAGE 93

SOURCE CODE: SAME CONTENT AS SOURCE LISTING BUT HAND ASSEMBLED.

OBJECT LISTING: SOFTWARE PROGRAM LISTING RESULTING FROM COMPUTER SOFTWARE CONTROLLED ASSEMBLY PROCESS THAT ONLY INCLUDES MACHINE READABLE OBJECT CODED INSTRUCTIONS AND MEMORY ADDRESS ASSIGNMENTS.

OBJECT CODE SAME CONTENT AS OBJECT LISTING BUT HAND ASSEMBLED.

HARD COPY: XEROX OR PRINTED COPY

CODE: HAND ASSEMBLED CODE (SOURCE, OBJECT, OR ASSEMBLY CODE).

LISTING: COMPUTER FORMATED LISTING.

DUMP: COMPUTER MEMORY DUMP.

MSD PROGRAMS

CPU TYPE	SYMBOLIC NAME	DESCRIPTIVE NAME	MSD # & MEDIA	PACK	REV #	PRICE IN $ +CALIF. TAX(*) +USA POSTAGE(:)
6502	APPLECD	6502 APPLE COMPUTER DISASSEMBLER BY ALLEN BAUM & STEPHEN WOZNIAK-INTERFACE AGE, SEPT. 1976, VOL.1,#10.	1-TEXT	<		2.00+0.12+1.00
			1-HCAL	<		INC. WITH TEXT
			1-PACK	*		
8080	LPTIIHF	LOAD 8080 PAPER TAPE IN INTEL HEX FORMAT BY BURT HASHIZUME-INTERFACE AGE, OCT. 1976, VOL.1,#11.	2-PTAL	<	0	5.00+0.30+1.00
			2-TEXT	<		2.00+0.12+1.00
			2-HCAL	<		INC. WITH TEXT
			2-PACK	*		
8080	BFWOA	8080 BINARY FILES WITH OPTIONAL AUTOSTART BY WILLIAM H. JORDAN-INTERFACE AGE, OCT. 1976, VOL.1,#11.	3-PTAL	<	0	5.00+0.30+1.00
			3-PTOD	<		INC. WITH PTAL
			3-TEXT	<		2.00+0.12+1.00
			3-HCAL	<		INC. WITH TEXT
			3-PACK	*		
6800	MINOPS	MIN OPERATING SYSTEM BY ED KEITH & DENNIS HESCOX-INTERFACE AGE, OCT. 1976, VOL.1,#11. PTAL+ INCLUDES OPERATING INSTRUCTIONS, PAPER TAPE FORMAT AND SAMPLE RUN	4-PTAL+	<	0	5.00+0.30+1.00
			4-PTOD	<		INC. WITH PTAL
			4-TEXT	<		2.00+0.12+1.00
			4-HCAL	<		INC. WITH TEXT
			4-PACK	*		
8080	DBBDP	DR. BEATTIE'S BASIC DIET PLANNING BY DR. BEATTIE-INTERFACE AGE, OCT. 1976, VOL.1,#11.	5-TEXT	<	0	2.00+0.12+1.00
			5-HCBL	<		INC. WITH TEXT
			5-PTBL	<		5.00+0.30+1.00
			5-PACK	*		
6800	EZMERPS	ECHO 1, ZERO MEMORY, ECHO REVERSE & PRINT SUBROUTINES BY HOWARD BERENBON-INTERFACE AGE, OCT. 1976, VOL.1,#11.	6-PTAL	<	0	3.00+0.18+1.00
			6-TEXT	<		1.00+0.06+1.00
			6-HCAL	<		INC. WITH TEXT
			6-PACK	*		
8080	ESP-1	ESP-1 SOFTWARE PACKAGE BY MICHAEL SHRAYER-INTERFACE AGE, OCT. 1976, VOL.1,#11. PTGR IS PAPER TAPE COPY OF GRAMMAR.	7-PTOD	<	%0	30.00+1.80+1.50
			7-MAN	*		INC. WITH PTOD
			7-CTOD	*		30.00+1.80+1.50
			7-MAN	*		INC. WITH CTOD
			7-PTGR	*		5.00+0.30+1.50
			7-TEXT	*		INC. WITH PTGR
			7-HCGR	*		INC. WITH MAN
			7-PACK	*		
8080	PTSP-1	PROCESSOR TECHNOLOGY SOFTWARE PACKAGE NO. 1 SUMMARY BY R. A. STEVENS-INTERFACE AGE, OCT. 1976, VOL.1,#11.	8-PTTL	<	%0	4.00+0.24+1.50
			8-TEXT	<		INC. WITH PTTL
			8-PACK	*		
8080	ERAMMT	EXHAUSTIVE 8080 RAM MEMORY TEST PROGRAM BY T.E.TRAVIS-INTERFACE AGE, NOV. 1976, VOL.1,#12.	9-PTAL	<	0	4.00+0.24+1.00
			9-PTOD	<		INC. WITH PTAL
			9-TEXT	<		1.00+0.06+1.00
			9-HCAL	<		INC. WITH TEXT
			9-HCOD	<		INC. WITH TEXT
			9-PACK	*		
6800	MEMDMP-1	SWTPC 6800 MEMORY DUMP PROGRAM MEMDMP-1 BY GARY KAY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	10-PTAL	<	0	3.00+0.18+1.00
			10-PTSL	<	0	3.00+0.18+1.00
			10-PTOD	<		INC. WITH PTSL
			10-TEXT	<		1.00+0.06+1.00
			10-HCAL	<		INC. WITH TEXT
			10-PACK	*		
6800	ROBIT-1	SWTPC 6800 ROTATING BIT RAM MEMORY DIAGNOSTIC PROGRAM ROBIT-1 BY GARY KAY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	11-PTAL	<	0	3.00+0.18+1.00
			11-PTSL	<	0	3.00+0.18+1.00
			11-PTOD	<		INC. WITH PTSL
			11-TEXT	<		1.00+0.06+1.00
			11-HCAL	<		INC. WITH TEXT
			11-PACK	*		
6800	MEMCON-1	SWTPC 6800 SHORT MEMORY ADDRESS CONVERGENCE PROGRAM MEMCON-1 BY GARY KAY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	12-PTAL	<	0	3.00+0.18+1.00
			12-PTSL	<	0	3.00+0.18+1.00
			12-PTOD	<		INC. WITH PTSL
			12-TEXT	<		1.00+0.06+1.00
			12-HCAL	<		INC. WITH TEXT
			12-PACK	*		
6800	BJIB	BLACKJACK IN BASIC PROGRAM BY ED KEITH & DENNIS HESCOX. THE BJIB PAPER TAPE OBJECT CODE REQUIRES ROBERT UITERWYK'S SWTPC MICROBASIC OPERATING SYSTEM-INTERFACE AGE, NOV. 1976, VOL.1,#12. PTBL+ INCLUDES SAMPLE RUN, INSTRUCTIONS, LIST OF VARIABLES AND LIST OF ROUTINES.	13-PTBL		0	5.00+0.30+1.00
			13-PTBL+	<		6.00+0.36+1.00
			13-TEXT	<		1.00+0.06+1.00
			13-HCBL	<		INC. WITH TEXT
			13-PACK	*		
6502	RFPR	REVISED FLOATING POINT ROUTINES FOR 6502* BY ROY RANKIN & STEVE WOZNIAK - INTERFACE AGE, NOV. 1976, VOL.1,#12. NOTE * - ORIGINAL MATH PACKAGE FIRST APPEARED IN DR. DOBB'S JOURNAL, AUG. 1976, VOL.1,#7. DR. DOBB'S JOURNAL, AUG. 1976, VOL.1,#7.	14-PTOD	<	1	3.00+0.18+1.00
			14-PTAL	<		9.00+0.54+1.50
			14-PTSL	<		9.00+0.54+1.50
			14-TEXT	<		2.00+0.12+1.00
			14-HCAL	<		INC. WITH TEXT
			14-PACK	*		
6800	HISPDMUP	HIGH SPEED DOUBLE PRECISON MULTIPLICATION SUBROUTINE-HISPDMUP BY PERMISSION AND COURTESY OF MOTOROLA'S M6800 USER GROUP LIBRARY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	15-PTAL	<	0	4.00+0.24+1.00
			15-TEXT	<		1.00+0.06+1.00
			15-HCAL	<		INC. WITH TEXT
			15-PACK	*		
6800	DIV16	REENTRANT 16 BIT DIVIDE SUBROUTINE - DIV16 BY PERMISSION AND COURTESY OF MOTOROLA'S M6800 USER GROUP LIBRARY- INTERFACE AGE, NOV. 1976, VOL.1,#12.	16-PTAL	<	1	4.00+0.24+1.00
			16-TEXT	<		1.00+0.06+1.00
			16-HCAL	<		INC. WITH TEXT
			16-PACK	*		
6800	RENTMUP	REENTRANT DOUBLE PRECISION MULTIPLICATION SUBROUTINE-RENTMUP BY PERMISSION AND COURTESY OF MOTOROLA'S M6800 USER GROUP LIBRARY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	17-PTAL	<	0	4.00+0.24+1.00
			17-TEXT	<		1.00+0.06+1.00
			17-HCAL	<		INC. WITH TEXT
			17-PACK	*		
8080	HOMEC	COMPUTER OR CONTROLLER BY TERRY RENSON, INTEL - INTERFACE AGE, SEPT. 1976, VOL.1,#10.	18-PTAL	<	0	3.00+0.18+1.00
			18-PTSL	<		3.00+0.18+1.00
			18-TEXT	<		1.00+0.06+1.00
			18-HCAL	<		INC. WITH TEXT
			18-PACK	*		
8080	LCST	STARTREK BY LYNN COCHRAN-INTERFACE, JUNE 1976, VOL.1,#7.	19-PTBL	<	0	5.00+0.30+1.00
			19-TEXT	<		1.00+0.06+1.00
			19-HCBL	<		INC. WITH TEXT
			19-PACK	*		
8080	WSPG	WORD SEARCH PUZZLE GENERATOR BY RICHARD S. EDELMAN - INTERFACE, JULY 1976, VOL.1,#8.	20-PTBL	<	0	4.00+0.24+1.00
			20-TEXT	<		1.00+0.06+1.00
			20-HCBL	<		INC. WITH TEXT
			20-PACK	*		
8080	PGBIORHY	BIORHYTHM BY PAUL GREEN - INTERFACE AGE, AUG. 1976, VOL.1,#9.	21-PTBL	<	0	4.00+0.24+1.00
			21-TEXT	<		1.00+0.12+1.00
			21-HCBL	<		INC. WITH TEXT
			21-PACK	*		
8080	WDBIORHY	BIORHYTHMS IN PRACTICE BY WILLIAM L. DONHAN, M.D. - INTERFACE AGE, AUG. 1976, VOL.1,#9.	22-PTBL	<	0	4.00+0.24+1.00
			22-TEXT	<		1.00+0.06+1.00
			22-HCBL	<		INC. WITH TEXT
			22-PACK	*		
8080	REBJ	BLACKJACK BY RICHARD S. EDELMAN - INTERFACE AGE, AUG. 1976, VOL.1,#9.	23-PTBL	<	0	5.00+0.30+1.00
			23-TEXT	<		1.00+0.06+1.00
			23-HCBL	<		INC. WITH TEXT
			23-PACK	*		
8080	BLUFF	BLUFF BY PHIL FELDMAN & TOM RUGE - INTERFACE AGE, SEPT. 1976, VOL.1,#10.	24-PTBL	<	0	5.00+0.30+1.00
			24-TEXT	<		1.00+0.06+1.00
			24-HCBL	<		INC. WITH TEXT
			24-PACK	*		
6800	RABSIMB	RELATIVE ADDRESS BACK-STEPPER IN MICRO-BASIC BY J. HUFFMAN - INTERFACE AGE, DEC. 1976, VOL.1,#13.	25-PTBL	<	0	3.00+0.18+1.00
			25-HCBL	<		1.00+0.06+1.00
			25-TEXT	<		INC. WITH HCBL
			25-PACK	*		
6800	TEFT6800	TEXT EDITOR FOR THE SWTPC-6800 BY MARK BORGERSON - INTERFACE AGE, DEC. 1976, VOL.1,#13. HCAL IS COPY OF FULL SIZE ASSEMBLY LISTING.	26-PTAL	<	0	10.00+0.36+2.00
			26-PTOD	<		5.00+0.30+1.25
			26-HCAL	<		3.00+0.18+1.50
			26-TEXT	<		2.00+0.12+1.25
			26-PACK	*		
8080	WPATBX	WANG'S PALO ALTO TINY BASIC BY ROGER RAUSKOLB - INTERFACE AGE, DEC. 1976, VOL.1,#13. HCAL & HCSL ARE COPIES OF FULL SIZE CODE	27-PTSL	<	0	10.00+0.60+2.00
			27-PTOD	<		5.00+0.30+1.50
			27-HCAL	<		4.00+0.24+1.50
			27-TEXT	<		INC. WITH HCAL
			27-HCSL	<		4.00+0.24+1.50
			27-PACK	*		
8080	LLLBI	LLL 8080 BASIC INTERPRETER GRAMMAR BY JERRY BARBER & ROYCE ECKARD - SUBMITTED BY E. R. FISHER - INTERFACE AGE, DEC. 1976,VOL.1,#13(PART 1). PART 2 PUBLISHED JAN. 1977, VOL.2,#1. TEXT1 IS PART 1, TEXT2 IS PART 2 AND HCAL2 IS FULL SIZE XEROX COPY OF BASIC ASSEMBLY LISTING WITHOUT FLOATING POINT.	28-TEXT1	<	0	2.00+0.12+1.25
			28-HCAL2	<		5.00+0.30+2.00
			28-TEXT2	<		2.00+0.12+1.25
			28-HCAL3	<		5.00+0.30+2.00
			28-TEXT3	<		2.00+0.12+1.25
SC/MP	NIBL	NIBL-NATIONAL'S TINY BASIC GRAMMAR FOR SC/MP BY PHIL ROYBAL - INTERFACE AGE, DEC. 1976, VOL.1,#13. ASSEMBLY LISTING PUBLISHED JAN. 1977, VOL.2,#1.	29-TEXT	<	0	2.00+0.12+1.25
			29-HCAL	<		10.00+3.00+2.00
			29-PTSL	<		10.00+3.00+2.00
			29-PTOD	<		5.00+1.50+1.00
			29-PTGR	<		2.00+0.12+1.00
			29-PACK	*		
SC/MP	MWBAGELS	BAGELS BY DR. MARVIN WINZINREAD BY PERMISSION & COURTESY OF NATIONAL SEMICONDUCTOR - INTERFACE	30-PTBL	<	0	2.00+0.12+1.00

BRANCH TO PAGE 120

BASIC Algorithms for Common Math Functions

By Michael P. Burton (BAFCMF)

Some smaller versions of BASIC utilize standard floating point arithmetic but do not contain the commonly used math functions sine, cosine, e^x, $\log^e$ and square root. All of these functions except square root can be expressed as series, which are easily converted to algorithms written in BASIC. For the square root function, there is an approximation method which can also be written as a BASIC algorithm. The following routines are the algorithms for sin(x), cos(x), exp(x), ln(x) and sqrt(x). Please note the constraints placed on the values of x. Within the given boundaries of x, the routines are accurate to seven digits.

The following BASIC program was developed for SWTP's 6800 Microcomputer and runs on the SWTP's standard 4K BASIC Interpreter version 2.0 with floating point.

INTERFACE EQUIPMENT; SWTPC CT1024 with SWTPC's AC-30 Cassette Interface.

SEE MICROCOMPUTER SOFTWARE DEPOSITARY PROGRAM INDEX FOR COPIES OF THIS PROGRAM

```
BASIC ALGORITHMS FOR COMMON MATH FUNCTIONS
SYMBOLIC NAME: BAFCMF
PROGRAMMER: MICHAEL P. BURTON
MICROCOMPUTER: SWTPC 6800
INTERFACE EQUIPMENT: SWTPC CT1024 WITH AC-30
                     CASSETTE INTERFACE
SUPPORT SOFTWARE: SWTPC 4K BASIC

1000 REM SIN ALGORITHM
1005 REM X=INPUT ANGLE EXPRESSED IN DEGREES
1010 REM X1=RESULT (X1=SIN(X))
1015 REM -180.0<=X<=180.0
1020 LET N1=1
1025 LET N2=-1
1030 LET X1=0.0
1035 LET X2=1.0
1040 LET X3=X*(3.14159265/180.0)
1045 IF X>=(-180.0) THEN IF X<=180.0 THEN 1055
1050 PRINT "SIN BOUND ERR"
1055 FOR N=1 TO 15
1060 LET N1=N1*N
1065 LET X2=X2*X3
1070 IF N=(INT(N/2)*2) THEN 1085
1075 LET N2=INT(N2*(-1))
1080 LET X1=X1+(X2*INT(N2))/(INT(N1))
1085 NEXT N
1090 RETURN
1100 REM COSINE ALGORITHM
1105 REM X=INPUT ANGLE EXPRESSED IN DEGREES
1110 REM X1=RESULT (X1=COS(X))
1115 REM -180.0<=X<=180.0
1120 LET N1=1
1125 LET N2=1
1130 LET X1=1.0
1135 LET X3=X*(3.14159265/180.0)
1140 LET X2=X3
1145 IF X>=(-180.0) THEN IF X<=180.0 THEN 1155
1150 PRINT "COS BOUND ERR"
1155 FOR N=2 TO 16
1160 LET N1=N1*N
1165 LET X2=X2*X3
1170 IF N<>(INT(N/2)*2) THEN 1185
1175 LET N2=INT(N2*(-1))
1180 LET X1=X1+(X2*INT(N2))/(INT(N1))
1185 NEXT N
1190 RETURN
1200 REM EXPONENTIAL ALGORITHM
1205 REM X=EXPONENT OF E
1210 REM X1=RESULT (X1=EXP(X))
1215 REM -5.0<=X<=22.0
1220 LET N1=1
1225 LET X1=1.0
1230 LET X2=1.0
1235 IF X>=(-5.0) THEN IF X<=22.0 THEN 1245
1240 PRINT "EXP BOUND ERR"
1245 FOR N=1 TO 51
1250 LET N1=N1*N
1255 LET X2=X2*X
1260 LET X1=X1+X2/(INT(N1))
1265 NEXT N
1270 RETURN
1300 REM LOGARITHM ROUTINE
1305 REM X1=LN(X)
1310 REM 0.1<=X<=10.0
1315 LET X1=0.0
1320 LET X2=1.0
1325 IF X>=0.1 THEN IF X<=10.0 THEN 1335
1330 PRINT "LN BOUND ERR"
1335 FOR N=1 TO 51
1340 LET X2=X2*((X-1.0)/(X+1.0))
1345 IF N=(INT(N/2)*2) THEN 1355
1350 LET X1=X1+X2/INT(N)
1355 NEXT N
1360 LET X1=2.0*X1
1365 RETURN
1400 REM SQUARE ROOT ALGORITHM
1405 REM X=INPUT VALUE
1410 REM X1=RESULT (X1=SQRT(X))
1415 REM X>0.0
1420 IF X>0.0 THEN 1435
1425 PRINT "SQRT BOUND ERROR"
1430 RETURN
1435 LET X1=X/2.0
1437 LET X3=0
1440 LET X2=(X/X1-X1)/2
1445 IF X2=0.0 THEN RETURN
1450 IF X2=X3 THEN RETURN
1455 LET X1=X1+X2
1457 LET X3=X2
1460 GOTO 1440
9999 END
```

BASIC FLOATING POINT MATH PACKAGE

Part 3 of LLL 8080 Basic Interpreter Program

By David Mead

and modified by Hal Brand and Frank Olken

INTRODUCTION

This article is Part 3 of a series of four articles covering the LLL 8080 BASIC Interpreter program released to the public domain by Lawrence Livermore Laboratories. It includes the BASIC Floating Point Math Package assembly listing.

```
8080 MACRO ASSEMBLER, VER 1.1   ERRORS = 0 PAGE 1

                       ;    COMPANION FILE FOR BAS80
                       ;    CONTAINS FLOATING POINT ROUTINES AND SOME
                       ;    UTILITY ROUTINES.
                       ;
                       ;////FLOATING POINT PACKAGE FOR THE MCS8
                       ;////BY DAVID MEAD
                       ;////MODIFIED BY HAL BRAND 9/6/74
                       ;////MODIFIED FOR 24 BIT MANTISSAS***********
                       ;////PLUS ADDED I/O CONVERSION ROUTINES
                       ;////NEW ROUTINE COMMENTS
                       ;////ARE PRECEDED BY /
                       ;////OTHER CHANGES ARE NOTED BY **
                       ;////MODIFIED BY FRANK OLKEN 6/28/75
                       ;
0900                        ORG 4400O
                       ;
0FFD                   OUTP    EQU     7775Q            ;LINK TO BASIC
0FFA                   INP     EQU     7772Q            ;LINK TO BASIC
00C0                   MINCH   EQU     300Q         ;MINIMUM CHARACTERISTIC WITH
003F                   MAXCH   EQU     77Q          ;MAXIMUM CHARACTERISTIC WITH
                       ;
                       ;******************************************************
                       ;    //// DIVIDE SUBROUTINE
                       ;******************************************************
                       ;
0900   CD 69 0C        LDIV:   CALL    CSIGN        ;COMPUTE SIGN OF RESULT
0903   CD DA 0A                CALL    ZCHK         ;CHECK IF DIVIDEND = ZERO
0906   C2 12 09                JNZ     DTST2        ;IF DIVIDEND .NE. 0 CHECK DI
0909   CD E2 0A                CALL    BCHK         ;CHECK FOR ZERO/ZERO
090C   CA AB 0B                JZ      INDFC        ;ZERO/ZERO = INDEFINITE
090F   C3 AF 0B                JMP     WZERC        ;ZERO/NONZERO = ZERO
0912   CD E2 0A        DTST2:  CALL    BCHK         ;COME HERE IF DIVIDEND .NE.
0915   CA 5B 0C                JZ      OFLWC        ;NONZERO/ZERO = OVERFLOW
                                                    ;IF WE GET HERE, THINGS LOOK
0918   50                      MOV     E,L          ;SAVE BASE IN E
0919   69                      MOV L,C              ;BASE?6 TO L
091A   CD 10 0B                CALL DCLR            ;CLEAR QUOTIENT MANTISSA SLOT
091D   6B                      MOV L,E              ;RESTORE BASE IN L
091E   CD 10 0C                CALL ENT1            ;DO FIRST CYCLE
0921   69                      MOV L,C              ;BASE ?6 TO L
0922   CD E9 0A                CALL DLST            ;MOVE QUOTIENT OVER ONE PLACE
0925   16 17                   MVI D,23             ;NUMBER OF ITERATIONS TO D
0927   6B              REP3:   MOV L,E
0928   CD 0A 0C                CALL ENT2
092B   15                      DCR D                ;DEC D
092C   CA 3B 09                JZ   GOCN
092F   7D                      MOV A,L
0930   69                      MOV L,C              ;BASE?6 TO L
0931   4F                      MOV C,A
0932   CD E9 0A                CALL DLST            ;MOVE QUOTIENT MANT OVER
0935   7D                      MOV A,L              ;CPTR TO A
0936   59                      MOV E,C              ;LPTR TO E
0937   4F                      MOV C,A              ;CPTR TO C
0938   C3 27 09                JMP REP3
                       ;
093B   CD E1 0B        GOCN:    CALL    ACPS            ;CHECK IF RESULT IS NOR
093E   FA 4D 09                JM   CRIN
0941   7D                      MOV A,L              ;LPTR TO A
0942   69                      MOV L,C              ;CPTR TO L
0943   4F                      MOV C,A              ;LPTR TO C
0944   CD F9 0A                CALL DLST            ;SHIFT QUOTIENT LEFT
0947   4D                      MOV C,L
0948   6B                      MOV L,E
0949   CD 39 0C                CALL    LDCP         ;COMPUTE THE CHARACTERISTIC
094C   C9                      RET
                       ;
094D   CD 4C 0B        CRIN:    CALL    CFCHE           ;GET A=CHAR(H,L), E=CHA
0950   93                      SUB     E            ;NEW CHAR = CHAR(DIVIDEND) -
0951   FE 7F                   CPI     177Q         ;CHECK MAX POSITIVE NUMBER
0953   CA 5B 0C                JZ      OFLWC        ;JUMP ON OVERFLOW
0956   C6 01                   ADI     1            ;ADD 1 SINCE WE DID NOT LEFT
0958   CD 44 0C                CALL    CCHK         ;CHECK AND STORE CHARACTERIS
095B   C9                      RET                  ;RETURN
                       ;
                       ;******************************************************
                       ;    //// ADDITION SUBROUTINE
                       ;******************************************************
                       ;
095C   AF              LADD:   XRA A                ;/***SET UP TO ADD
095D   C3 62 09                JMP LADS             ;/NOW DO IT
                       ;
                       ;******************************************************
                       ;    //// SUBTRACTION SUBROUTINE
                       ;******************************************************
                       ;
0960   3E 80           LSUB:   MVI A,200Q           ;/****SET UP TO SUBTRACT
                       ;                 SUBROUTINE LADS
                       ;                 FLOATING POINT ADD OR SUB
                       ;                 A?128 ON ENTRY?SUB
                       ;                 A?0 ON ENTRY?ADD
                       ;                 F-S?F,FIRST OPER DESTROYED
                       ;                 BASE ?11 USED FOR SCRATCH
0962   CD EF 0B        LADS:   CALL ACPR            ;SAVE ENTRY PNT AT BASE ?6
0965   CD E2 0A                CALL    BCHK         ;CHECK ADDEND/SUBTRAHEND = Z
0968   C8                      RZ                   ;IF SO, RESULT=ARG SO RETURN
                                                    ;THIS WILL PREVENT UNDERFLOW
                                                    ;ZERO + OR - ZERO
0969   CD 5B 0B                CALL CCMP
096C   CA 9C 09                JZ   EQ02            ;IF EQUAL, GO ON
096F   57                      MOV D,A              ;SAVE LPTR CHAR IN D
0970   DA 7F 09                JC   LLTB
0973   93                      SUB E                ;L.GT.B IF HERE
0974   E6 7F                   ANI 127
0976   57                      MOV D,A              ;DIFFERENCE TO D
0977   5D                      MOV E,L              ;SAVE BASE IN E
0978   69                      MOV L,C              ;C PTR TO L
0979   2C                      INR L                ;C PTR?1 TO L
097A   73                      MOV M,E              ;SAVE BASE IN C PTR?1
097B   68                      MOV L,B              ;B PTR TO L
097C   C3 84 09                JMP NCHK
097F   78              LLTB:   MOV A,B              ;L.LT.B IF HERE,BPTR TO A
0980   92                      SUB D                ;SUBTRACT LPTR CHAR FROM BPTR
0981   E6 7F                   ANI 127
0983   57                      MOV D,A              ;DIFFERENCE TO D
0984   3E 18           NCHK:   MVI A,24
0986   BA                      CMP D
0987   D2 8C 09                JNC SH10
098A   16 18                   MVI D,24
098C   B7              SH10:   ORA A
098D   CD F8 0A                CALL DRST
0990   15                      DCR D
0991   C2 8C 09                JNZ SH10
0994   7D              EQUL:   MOV A,L
0995   B8                      CMP B
0996   C2 9C 09                JNZ EQ02             ;F.GT.S IF L.NE.B
0999   69                      MOV L,C              ;C PTR TO L
099A   2C                      INR L                ;C PTR?1 TO L
099B   6E                      MOV L,M              ;RESTORE L
099C   CD 02 0A        EQ02:   CALL LASD            ;CHECK WHAT TO
099F   CD EF 0B                CALL ACPR            ;SAVE ANSWER
09A2   FE 02                   CPI 2                ;TEST FOR ZERO ANSWER
09A4   C2 AA 09                JNZ NOT0
09A7   C3 BD 0B                JMP        WZER          ;WRITE FLOATING ZERO AND RET
                       ;
09AA   16 01           NOT0:   MVI D,1              ;WILL TEST FOR SUB
09AC   A2                      ANA D
09AD   CA D6 09                JZ   ADDZ            ;LSB?1 IMPLIES SUB
09B0   CD E7 0B                CALL TSTR            ;CHECK NORMAL/REVERSE
09B3   CA B9 09                JZ   SUBZ            ;IF NORMAL,GO SUBZ
09B6   7D                      MOV A,L              ;OTHERWISE REVERSE
09B7   68                      MOV L,B              ;ROLES
09B8   47                      MOV B,A              ;OF L AND B
                       ;
09B9   CD 26 0B        SUBZ:   CALL    DSUB            ;SUBTRACT SMALLER FROM BIGGE
09BC   CD EF 09                CALL    MANT            ;SET UP SIGN OF RESULT
09BF   CD E7 0B                CALL    TSTR            ;SEE IF WE NEED TO INTERCHAN
                                                       ;BPTR AND LPTR
09C2   CA AD 0A                JZ      NORM            ;NO INTERCHANGE NECESSARY, S
                                                       ;AND RETURN
09C5   7D                      MOV A,L              ;INTERCHANGE
09C6   68                      MOV L,B              ;L
09C7   47                      MOV B,A              ;AND B
09C8   79                      MOV A,C              ;CPTR TO A
09C9   48                      MOV C,B              ;BPTR TO C
09CA   5D                      MOV E,L              ;LPTR TO E
09CB   47                      MOV B,A              ;CPTR TO B
09CC   CD 24 0C                CALL LXFR            ;MOVE ?BPTR> TO ?LPTR>
09CF   78                      MOV A,B
09D0   41                      MOV B,C
09D1   4F                      MOV C,A
09D2   6B                      MOV L,E
09D3   C3 AD 0A                JMP     NORM            ;NORMALIZE RESULT AND RETURN
                       ;
                       ;  COPY THE LARGER CHARACTERISTIC TO THE RESULT
                       ;
09D6   CD 5B 0B        ADDZ:   CALL    CCMP            ;COMPARE THE CHARACTERISTICS
09D9   D2 DF 09                JNC     ADD2            ;IF CHAR(H,L) .GE. CHAR(H,B)
09DC   CD 8D 0C                CALL    BCTL            ;IF CHAR(H,L) .LT. CHAR(H,B)
                                                       ;CHAR(H,B) TO CHAR(H,L)
09DF   CD EF 09        ADD2:   CALL    MANT            ;COMPUTE SIGN OF RESULT
09E2   CD 06 0B                CALL    DADD            ;ADD MANTISSAS
09E5   D2 02 0B                 JNC    SCCFG              ;IF THERE IS NO OVFLW -
09E8   CD F8 0A                CALL    DRST            ;IF OVERFLOW SHIFT RIGHT
09EB   CD B6 0B                CALL    INCR            ;AND INCREMENT CHARACTERISTI
09EE   C9                      RET                     ;ALL DONE, SO RETURN
                       ;
                       ;  THIS ROUTINE STORES THE MANTISSA SIGN IN THE RESULT
                       ;  THE SIGN HAS PREVIOUSLY BEEN COMPUTED BY LASD.
                       ;
09EF   5D              MANT:   MOV E,L              ;SAVE L PTR
09F0   69                      MOV L,C              ;C PTR TO L
09F1   7E                      MOV A,M              ;LOAD INDEX WORD
09F2   E6 80                   ANI 128              ;SCARE SIGN
09F4   6B                      MOV L,E              ;RESTORE L PTR
09F5   2C                      INR L                ;L PTR?2
09F6   2C                      INR L
09F7   2C                      INR L                ;TO L
09F8   5F                      MOV E,A              ;SAVE SIGN IN E
09F9   7E                      MOV A,M
09FA   E6 7F                   ANI 127              ;SCARE CHAR
09FC   83                      ADD E                ;ADD SIGN
09FD   77                      MOV M,A              ;STORE IT
09FE   2D                      DCR L                ;RESTORE
09FF   2D                      DCR L
0A00   2D                      DCR L                ;L PTR
0A01   C9                      RET
                       ;
                       ;                 SUBROUTINE LASD
                       ;                 UTILITY ROUTINE FOR LADS
                       ;                 CALCULATES TRUE OPER AND SGN
                       ;                 RETURNS ANSWER IN
0A02   CD 79 0C        LASD:   CALL MSFH            ;FETCH MANT SIGNS, F IN A,D
0A05   BB                      CMP E                ;COMPARE SIGNS
0A06   DA 34 0A                JC   ABCH            ;F?,S- MEANS GO TO A BRANCH
0A09   C2 3D 0A                JNZ  BBCH            ;F- S? MEANS GO TO B BRANCH
0A0C   83                      ADD E                ;SAME SIGN IF HERE, ADD SIGNS
0A0D   DA 22 0A                JC   BMIN            ;IF BOTH MINUS, WILL OVERFLOW
0A10   CD E1 0B                CALL AORS            ;BOTH POS IF HERE
0A13   F2 46 0A                JP   L000            ;IF AN ADD, LOAD 0
0A16   CD F4 0B        COM1:   CALL DCMP            ;COMPARE F WITH S
0A19   DA 54 0A                JC   L131            ;S.GT.F,SO LOAD 131
0A1C   C2 48 0A                JNZ  L001            ;F.GT.S,SO LOAD 1
0A1F   3E 02           L002:   MVI A,2              ;ERROR CONDITION, ZERO ANSWER
0A21   C9                      RET
0A22   CD E1 0B        BMIN:   CALL AORS            ;CHECK FOR ADD OR SUB
0A25   F2 4F 0A                JP   L128            ;ADD, SO LOAD 128
0A28   CD F4 0B        COM2:   CALL DCMP            ;COMPARE F WITH S
0A2B   DA 4B 0A                JC   L003            ;S.GT.F,SO LOAD 3
0A2E   C2 51 0A                JNZ  L129            ;FGT.S.SO LOAD 129
0A31   C3 1F 0A                JMP  L002            ;ERROR
0A34   CD E1 0B        ABCH:   CALL AORS            ;FT,S- SO TEST FOR A/S
0A37   FA 46 0A                JM   L000            ;SUBTRACT, SO LOAD 0
0A3A   C3 16 0A                JMP  COM1            ;ADD, SO GO TO DCMP
0A3D   CD E1 0B        BBCH:   CALL AORS            ;F-,S?, SO TEST FOR A/S
0A40   FA 4F 0A                JM   L128            ;SUB
0A43   C3 28 0A                JMP  COM2            ;ADD
0A46   AF              L000:   XRA A
0A47   C9                      RET
0A48   3E 01           L001:   MVI A,1
0A4A   C9                      RET
0A4B   3E 03           L003:   MVI A,3
```

```
0A40  C9              RET
0A4E  3E 80     L128:  MVI A,128
0A50  C9              RET
0A51  3E 81     L129:  MVI A,129
0A53  C9              RET
0A54  3E 83     L131:  MVI A,131
0A56  C9              RET

                       SUBROUTINE LMCM
                       COMPARES THE MAGNITUDE OF
                       TWO FLOATING PNT NUMBERS
                       Z?1 IF ?,C?1 IF F.LT.S.
0A57  CD 5B 08  LMCM:  CALL CCMP      ;CHECK CHARS
0A5A  C0              RNZ            ;RETURN IF NOT EQUAL
0A5B  CD F4 08        CALL DCMP      ;IF EQUAL, CHECK MANTS
0A5E  C9              RET

                ******************************************
                       //// MULTIPLY SUBROUTINE
                ******************************************

                       SUBROUTINE LMUL
                       FLOATING POINT MULTIPLY
                       L PTR X B PTR TO C PTR

0A5F  CD 69 0C  LMUL:  CALL   CSIGN   ;COMPUTE SIGN OF RESULT AND
0A62  CD DA 0A         CALL   ZCHK    ;CHECK FIRST OPERAND FOR ZER
0A65  CA AF 0B         JZ     WZERC   ;ZERO * ANYTHING = ZERO
0A68  CD E2 0A         CALL   BCHK    ;CHECK SECOND OPERAND FOR ZE
0A6B  CA AF 0B         JZ     WZERC   ;ANYTHING * ZERO = ZERO
0A6E  5D               MOV E,L        ;SAVE L PTR
0A6F  69               MOV L,C        ;C PTR TO L
0A70  CD 1D 0B         CALL DCLR      ;CLR PRODUCT MANT LOCS
0A73  6B               MOV L,E        ;L PTR TO L
0A74  16 18            MVI D,24       ;LOAD NUMBER ITERATIONS
0A76  CD F8 0A  KPGO:  CALL DRST      ;SHIFT L PTR RIGHT
0A79  DA A4 0A         JC   MADD      ;WILL ADD B PTR IF C?1
0A7C  7D               MOV A,L        ;INTERCHANGE
0A7D  69               MOV L,C        ;L AND
0A7E  4F               MOV C,A        ;C PTRS
0A7F  CD F8 0A  INTR:  CALL DRST      ;SHIFT PRODUCT OVER
0A82  7D               MOV A,L        ;INTERCHANGE
0A83  69               MOV L,C        ;L AND C PTRS?BACK TO
0A84  4F               MOV C,A        ;ORIGINAL>
0A85  15               DCR D
0A86  C2 76 0A         JNZ KPGO       ;MORE CYCLES IF Z?0
0A89  CD E1 0B         CALL    ADRS   ;TEST IF RESULT IS NORMALIZE
0A8C  FA 40 0C         JM      LMCP   ;IF NORMALIZED GO COMPUTE CH
0A8F  5D               MOV     E,L    ;SAVE LPTR IN E
0A90  69               MOV     L,C    ;SET L=CPTR
0A91  CD E9 0A         CALL    DLST   ;LEFT SHIFT RESULT TO NORMAL
0A94  6B               MOV     L,E    ;RESTORE LPTR
0A95  CD 4C 0B         CALL    CFCHE  ;OTHERWISE SET A=CHAR(H,L),
0A98  83               ADD     E      ;CHAR(RESULT) = CHAR(H,L) +
0A99  FE 80            CPI     200Q   ;CHECK FOR SMALLEST NEGATIVE
0A9B  CA 62 0C         JZ      UFLWC  ;IF SO THEN UNDERFLOW
0A9E  D6 01            SUI     1      ;SUBTRACT 1 TO COMPENSATE FO
0AA0  CD 44 0C         CALL    CCHK   ;CHECK CHARACTERISTIC AND ST
0AA3  C9               RET            ;RETURN

0AA4  7D        MADD:  MOV A,L        ;INTERCHANGE
0AA5  69               MOV L,C        ;L AND
0AA6  4F               MOV C,A        ;C PTRS
0AA7  CD 06 0B         CALL DADD      ;ACCUMULATE PRODUCT
0AAA  C3 7F 0A         JMP INTR

                SUBROUTINE NORM

                   THIS SUBROUTINE WILL NORMALIZE A FLOATING POINT
                   NUMBER, PRESERVING ITS ORIGINAL SIGN.
                   WE CHECK FOR UNDERFLOW AND SET THE CONDITION
                   FLAG APPROPRIATELY.  (SEE ERROR RETURNS).
                   THER IS AN ENTRY POINT TO FLOAT A SIGNED INTEGER
                   (FLOAT) AND AN ENTRY POINT TO FLOAT AN UNSIGNED
                   INTEGER.

                ENTRY POINTS:

                   NORM  - NORMALIZE FLOATING PT NUMBER AT (H,L)
                   FLOAT - FLOAT TRIPLE PRECISION INTEGER AT (H,L)
                           PRESERVING SIGN BIT IN (H,L)+3
                   DFXL  - FLOAT UNSIGNED (POSITIVE) TRIPLE PRECISIO
                           AT (H,L)

                ;REGISTERS ON EXIT:

                   A = CONDITION FLAG (SEE ERROR RETURNS)
                   D,E = GARBAGE
                   B,C,H,L = SAME AS ON ENTRY

0AAD  5D        NORM:  MOV    E,L     ;SAVE L IN E
0AAE  CD 41 0B  NORML: CALL   GCHAR   ;GET CHAR(H,L) IN A WITH SIG
0AB1  57               MOV    D,A     ;SAVE CHAR IN D
0AB2  6B        FXL1:  MOV    L,E     ;RESTORE L
0AB3  CD DA 0A  FXL2:  CALL   ZMCHK   ;CHECK FOR ZERO MANTISSA
0AB6  CA BD 0B         JZ     WZER    ;IF ZERO MANTISSA THEN ZERO
0AB9  7E        REP6:  MOV    A,M     ;GET MOST SIGNIFICANT BYTE O
                                      ;MANTISSA
0ABA  B7               ORA    A       ;SET FLAGS
0ABB  FA C8 0A         JM     SCHAR   ;IF MOST SIGNIFICANT BIT = 1
                                      ;NUMBER IS NORMALIZED AND WE
                                      ;STORE THE CHARACTERISTIC
0ABE  7A               MOV    A,D     ;OTHERWISE CHECK FOR UNDERFL
0ABF  FE C0            CPI    MINCH   ;COMPARE WITH MINIMUM CHAR
0AC1  CA 63 0B         JZ     WUND    ;IF EQUAL THEN UNDERFLOW
0AC4  CD E9 0A         CALL   DLST    ;SHIFT MANTISSA LEFT
0AC7  15               DCR    D       ;DECREMENT CHARACTERSTIC
0AC8  C3 B9 0A         JMP    REP6    ;LOOP AN TEST NEXT BIT
0ACB  C3 C3 0B  SCHAR: JMP    INCR3   ;STORE THE CHARACTERISTIC US
                                      ;THE SAME CODE AS THE INCREM

0ACE  5D        DFXL:  MOV    E,L     ;ENTER HERE TO FLOAT UNSIGNE
                                      ;INTEGER
                                      ;FIRT SAVE L IN E
0ACF  2C               INR    L       ;MAKE (H,L) POINT TO CHAR
0AD0  2C               INR    L       ;MAKE (H,L) POINT TO CHAR
0AD1  2C               INR    L       ;MAKE (H,L) POINT TO CHAR
0AD2  AF               XRA    A       ;ZERO ACCUMULATOR
0AD3  77               MOV    M,A     ;STORE A PLUS (+) SIGN
0AD4  6B               MOV    L,E     ;RESTORE L
0AD5  16 18     FLOAT: MVI    D,24    ;ENTER HERE TO FLOAT INTEGER
                                      ;PRESERVING ORIGINAL SIGN IN
                                      ;SET UP CHARACTERISTIC
0AD7  C3 B3 0A         JMP    FXL2    ;GO FLOAT THE NUMBER

                SUBROUTINE ZCHK

                   THIS ROUTINE SETS THE ZERO FLAG IF IT DETECTS
                   A FLOATING ZERO AT (H,L).

                SUBROUTINE ZMCHK

                   THIS ROUTINE SETS THE ZERO FLAG IF IT DETECTS A
                   ZERO MANTISSA AT (H,L)

                ZCHK:
0ADA  2C        ZMCHK: INR    L       ;SET L TO POINT LAST BYTE OF
0ADB  2C               INR    L       ;SET L TO POINT TO LAST BYTE
0ADC  7E               MOV    A,M     ;LOAD LEAST SIGNIFICANT BYTE
0ADD  2D               DCR    L       ;L POINTS TO MIDDLE BYTE
0ADE  B6               ORA    M       ;OR WITH LEAST SIGNFICANT BY
0ADF  2D               DCR    L       ;L POINTS TO MOST SIGNIFICANT
                                      ;OF MANTISSA (ORIGINAL VALUE
0AE0  B6               ORA    M       ;OR IN MOST SIGNIFICANT BYTE
0AE1  C9               RET            ;RETURNS WITH ZERO FLAG SET

                SUBROUTINE BCHK

                   THIS ROUTINE CHECKS (H,B) FOR FLOATING PT ZERO

0AE2  5D        BCHK:  MOV    E,L     ;SAVE LPTR IN E
0AE3  68               MOV    L,B     ;SET L=BPTR
0AE4  CD DA 0A         CALL   ZCHK    ;CHECK FOR ZERO
0AE7  6B               MOV    L,E     ;RESTORE L=LPTR
0AE8  C9               RET            ;RETURN

                       SUBROUTINE DLST
                       SHIFTS DBL WORD ONE PLACE LF

0AE9  2C        DLST:  INR L
0AEA  2C               INR L          ;/***TP
0AEB  7E               MOV A,M        ;LOAD IT
0AEC  B7               ORA A          ;KILL CARRY
0AED  17               RAL            ;SHIFT IT LEFT
0AEE  77               MOV M,A        ;STORE IT
0AEF  2D               DCR L
0AF0  7E               MOV A,M        ;LOAD IT
0AF1  17               RAL            ;SHIFT IT LEFT
                       IF CARRY SET BY FIRST SHIFT
                       IT WILL BE IN LSB OF SECOND
0AF2  77               MOV M,A
0AF3  2D               DCR L          ;/***TP EXTENSION
0AF4  7E               MOV A,M
0AF5  17               RAL
0AF6  77               MOV M,A        ;/***ALL DONE TP
0AF7  C9               RET
                       SUBROUTINE DRST
                       SHIFTS DOUBLE WORD ONE PLACE
                       TO THE RIGHT
                       DOES NOT AFFECT D
0AF8  5D        DRST:  MOV E,L        ;/***TP MODIFIED RIGHT SHIFT T
0AF9  7E               MOV A,M        ;LOAD FIRST WORD
0AFA  1F               RAR            ;ROTATE IT RIGHT
0AFB  77               MOV M,A        ;STORE IT
0AFC  2C               INR L          ;/*** TP
0AFD  7E               MOV A,M        ;LOAD SECOND WORD
0AFE  1F               RAR            ;SHIFT IT RIGHT
0AFF  77               MOV M,A        ;STORE IT
0B00  2C               INR L          ;/*** TP EXTENSION
0B01  7E               MOV A,M
0B02  1F               RAR
0B03  77               MOV M,A
0B04  6B               MOV L,E        ;/***TP - ALL DONE TP
0B05  C9               RET

                       SUBROUTINE DADD
                       ADDS TWO DOUBLE PRECISION
                       WORDS, C?1 IF THERE IS OVRFLW
0B06  5D        DADD:  MOV E,L        ;SAVE BASE IN E
0B07  68               MOV L,B        ;BASE 23 TO L
0B08  2C               INR L          ;BASE 24 TO L
0B09  2C               INR L          ;/***TP
0B0A  7E               MOV A,M        ;LOAD S MANTB
0B0B  6B               MOV L,E        ;BASE TO L
0B0C  2C               INR L          ;BASE 21 TO L
0B0D  2C               INR L          ;/***TP
0B0E  86               ADD M          ;ADD TWO MANTB?S
0B0F  77               MOV M,A        ;STORE ANSWER
0B10  68               MOV L,B        ;/***TP EXTENSION
0B11  2C               INR L
0B12  7E               MOV A,M
0B13  6B               MOV L,E
0B14  2C               INR L
0B15  8E               ADC M
0B16  77               MOV M,A        ;/***TP - ALL DONE
0B17  68               MOV L,B        ;BASE 23 TO L
0B18  7E               MOV A,M        ;MANTA OF S TO A
0B19  6B               MOV L,E        ;BASE TO L
0B1A  8E               ADC M          ;ADD WITH CARRY
0B1B  77               MOV M,A        ;STORE ANSWER
0B1C  C9               RET
                       SUBROUTINE DCLR
                       CLEARS TWO SUCCESSIVE
                       LOCATIONS OF MEMORY
0B1D  AF        DCLR:  XRA A
0B1E  77               MOV M,A
0B1F  2C               INR I
0B20  77               MOV M,A
0B21  2C               INR L          ;/***TP EXTENSION
0B22  77               MOV M,A        ;/***TP ZERO 3
0B23  2D               DCR L          ;/***TP - ALL DONE
0B24  2D               DCR L
0B25  C9               RET
                       /*****ALL NEW DSUB - SHORTER***
                       SUBROUTINE DSUB
                       DOUBLE PRECISION SUBTRACT
0B26  5D        DSUB:  MOV E,L        ;SAVE BASE IN E
0B27  2C               INR L          ;/***TP EXTENSION
0B28  2C               INR L          ;/START WITH LOWS
0B29  7E               MOV A,M        ;/GET ARG
0B2A  68               MOV L,B        ;/NOW SET UP TO SUB
0B2B  2C               INR L
0B2C  2C               INR L
0B2D  96               SUB M          ;/NOW DO IT
0B2E  6B               MOV L,E        ;/NOW MUST PUT IT BACK
0B2F  2C               INR L
0B30  2C               INR L
0B31  77               MOV M,A        ;/PUT BACK
0B32  2D               DCR L          ;/***TP - ALL DONE
0B33  7E               MOV A,M        ;/GET LOW OF LOP
0B34  68               MOV L,B        ;/SET TO BOP
0B35  2C               INR L          ;/SET TO BOP LOW
0B36  9E               SBB M          ;/GET DIFF. OF LOWS
0B37  6B               MOV L,E        ;/SAVE IN LOP LOW
0B38  2C               INR L          ;/TO LOP LOW
0B39  77               MOV M,A        ;/INTO RAM
0B3A  2D               DCR L          ;/BACK UP TO LOP HIGH
0B3B  7E               MOV A,M        ;/GET LOP HIGH
0B3C  68               MOV L,B        ;/SET TO BOP HIGH
0B3D  9E               SBB M          ;/SUB. WITH CARRY
0B3E  6B               MOV L,E        ;/SAVE IN LOP HIGH
0B3F  77               MOV M,A        ;/INTO RAM
0B40  C9               RET            ;/ALL DONE - MUCH SHORTER
                SUBROUTINE GCHAR

                   THIS SUBROUTINE RETURNS THE CHARACTERISTIC OF
                   THE FLOATING POINT NUMBER POINTED TO BY (H,L)
                   IN THE A REGISTER WITH ITS SIGN EXTENDED INTO THE
                   LEFTMOST BIT.

                REGISTERS ON EXIT:

                   A = CHARACTERISTIC OF (H,L) WITH SIGN EXTENDED
                   L = (ORIGINAL L) + 3
                   B,C,D,E,H = SAME AS ON ENTRY

0B41  2C        GCHAR: INR     L      ;MAKE (H,L) POINT TO CHAR
0B42  2C               INR     L      ;MAKE (H,L) POINT TO CHAR
0B43  2C               INR     L      ;MAKE (H,L) POINT TO CHAR
0B44  7E               MOV     A,M    ;SET A=CHAR + MANTISSA SIGN
0B45  E6 7F            ANI     177Q   ;GET RID OF MANTISSA SIGN BI
0B47  C6 40            ADI     100Q   ;PROPAGATE CHAR SIGN INTO LE
0B49  EE 40            XRI     100Q   ;RESTORE ORIGINAL CHAR SIGN
0B4B  C9               RET            ;RETURN WITH (H,L) POINTING
                                      ;CHAR = ORIGINAL (H,L)+3
                                      ;SOMEONE ELSE WILL CLEAN UP

                SUBROUTINE CFCHE

                   THIS SUBROUTINE RETURNS THE CHARACTERISTICS OF TH
                   FLOATING POINT NUMBERS POINTED TO BY (H,L) AND
                   (H,B) IN THE A AND E REGISTERS RESPECTIVELY,
                   WITH THEIR SIGNS EXTENDED INTO THE LEFTMOST BIT.

                REGISTERS ON EXIT:

                   A = CHARACTERISTIC OF (H,L) WITH SIGN EXTENDED
                   E = CHARACTERISTIC OF (H,B) WITH SIGN EXTENDED
                   B,C,H,L = SAME AS ON ENTRY
                   D = A

0B4C  5D        CFCHE: MOV     E,L    ;SAVE LPTR IN E
0B4D  68               MOV     L,B    ;SET L = BPTR
0B4E  CD 41 0B         CALL    GCHAR  ;GET CHAR(H,B) WITH SIGN EXT
0B51  6B               MOV     L,E    ;RESTORE L = LPTR
0B52  5F               MOV     E,A    ;SET E=CHAR(H,B) WITH SIGN E
0B53  CD 41 0B         CALL    GCHAR  ;SET A=CHAR(H,L) WITH SIGN E
0B56  2D               DCR     L      ;RESTORE L = LPTR
0B57  2D               DCR     L      ;RESTORE L = LPTR
0B58  2D               DCR     L      ;RESTORE L = LPTR
0B59  57               MOV     D,A    ;SET D=A=CHAR(H,L) WITH SIGN
0B5A  C9               RET

                SUBROUTINE CCMP

                   THIS SUBROUTINE COMPARES THE CHARACTERISTICS OF
                   FLOATING POINT NUMBERS POINTED TO BY (H,L) AND (H
                   THE ZERO FLIP-FLOP IS SET IF CHAR(H,L) EQUALS
                   CHAR(H,B).  IF CHAR(H,L) IS LESS THAN CHAR(H,B) T
                   THE CARRY BIT WILL BE SET.

                REGISTERS ON EXIT:

                   A = CHARACTERISTIC OF (H,L) WITH SIGN EXTENDED
                   E = CHARACTERISTIC OF (H,B) WITH SIGN EXTENDED
                   D = A
                   B,C,H,L = SAME AS ON ENTRY

0B5B  CD 4C 0B  CCMP:  CALL    CFCHE  ;FETCH CHARACTERISTICS WITH
                                      ;INTO A (CHAR(H,L)) AND E (C
0B5E  57               MOV     D,A    ;SAVE CHAR (H,L)
0B5F  93               SUB     E      ;SUBTRACT E (CHAR(H,B))
0B60  17               RAL            ;ROTATE SIGN BIT INTO CARRY
0B61  7A               MOV     A,D    ;RESTORE A=CHAR(H,L)
0B62  C9               RET            ;RETURN

                ERROR RETURNS

                   THE FOLLOWING CODE IS USED TO RETURN VARIOUS
                   ERROR CONDITIONS.  IN EACH CASE A FLOATING POINT
                   NUMBER IS STORED IN  THE 4 WORDS POINTED TO BY (H
                   AND A FLAG IS STORED IN THE ACCUMULATOR.

                   CONDITION    FLAG  RESULT (+)        RESULT (-)

                   UNDERFLOW    377   000 000 000 100   000 000 000
                   OVERFLOW     177   377 377 377 077   377 377 377
                   INDEFINITE   077   377 377 377 077   377 377 377
                   NORMAL       000   XXX XXX XXX XXX   XXX XXX XXX
                   NORMAL ZERO  000   000 000 000 100   (ALWAYS RETU

                ENTRY POINTS:

                   WUND - WRITE UNDERFLOW
                   WOVR - WRITE OVERFLOW
                   WIND - WRITE INDEFINITE
                   WZER - WRITE NORMAL ZERO

                                WRITE UNDERFLOW
```

```
0863   16 40       WUND:    MVI     D,100Q        ;LOAD CHARACTERISTIC INTO D R
0865   CD 9F 0B             CALL    WCHAR         ;WRITE CHARACTERISTIC
0868   3E 00       UFLW1:   MVI     A,0           ;LOAD MANTISSA VALUE
                                                  ;WE ASSUME HERE THAT ALL BYT
                                                  ;OF MANTISSA ARE THE SAME
086A   CD 98 0B             CALL    WMANT         ;WRITE THE MANTISSA
086D   3E FF                MVI     A,377Q        ;SET ACCUMULATOR TO FLAG
086F   B7                   ORA     A             ;SET FLAGS PROPERLY
0870   C9                   RET                   ;RETURN (WMANT RESTORED (H,L

                   ;        WRITE OVERFLOW
0871   16 3F       WOVR:    MVI     D,77Q         ;LOAD CHARACTERISTIC INTO D RE
0873   CD 9F 0B             CALL    WCHAR         ;WRITE CHARACTERISTIC
0876   3E FF       UFLW1:   MVI     A,377Q        ;LOAD MANTISSA VALUE
                                                  ;WE ASSUME HERE THAT ALL BYT
                                                  ;OF MANTISSA ARE THE SAME
0878   CD 98 0B             CALL    WMANT         ;WRITE THE MANTISSA
087B   3E 7F                MVI     A,177Q        ;SET ACCUMULATOR TO FLAG
087D   B7                   ORA     A             ;SET FLAGS PROPERLY
087E   C9                   RET                   ;RETURN (WMANT RESTORED (H,L

                   ;        WRITE INDEFINITE
087F   16 3F       WIND:    MVI     D,77Q         ;LOAD CHARACTERISTIC INTO D RE
0881   CD 9F 0B             CALL    WCHAR         ;WRITE CHARACTERISTIC
0884   3E FF       INDFI:   MVI     A,377Q        ;LOAD MANTISSA VALUE
                                                  ;WE ASSUME HERE THAT ALL BYT
                                                  ;OF MANTISSA ARE THE SAME
0886   CD 98 0B             CALL    WMANT         ;WRITE THE MANTISSA
0889   3E 3F                MVI     A,77Q         ;SET ACCUMULATOR TO FLAG
088B   B7                   ORA     A             ;SET FLAGS PROPERLY
088C   C9                   RET                   ;RETURN (WMANT RESTORED (H,L

088D   23          WZERT:   INX     H             ;WRITE NORMAL ZERO
088E   23                   INX     H
088F   23                   INX     H
0890   36 40                MVI     M,100Q        ;STORE CHARACTERISTIC FOR ZE
0892   AF                   XRA     A             ;ZERO ACCUMULATOR
0893   CD 98 0B             CALL    WMANT         ;STORE ZERO MANTISSA
0896   B7                   ORA     A             ;SET FLAGS PROPERLY
0897   C9                   RET                   ;RETURN

                   ;        ROUTINE TO WRITE MANTISSA FOR ERROR RETURNS
0898   2B          WMANT:   DCX     H             ;POINT TO LEAST SIGNIFI
                                                  ;OF MANTISSA
0899   77                   MOV     M,A           ;STORE LSBYTE OF MANTISSA
089A   2B                   DCX     H             ;PNT TO NEXT LEAST SIGN
                                                  ;OF MANTISSA
089B   77                   MOV     M,A           ;STORE NLSBYTE OF MANTISSA
089C   2B                   DCX     H             ;POINT TO MOST SIGNIFIC
                                                  ;OF MANTISSA
089D   77                   MOV     M,A           ;STORE MSBYTE OF MANTISSA
089E   C9                   RET                   ;RETURN (H,L) POINTS TO BEGI
                                                  ;FLOATING POINT RESULT
                   ;        ROUTINE TO WRITE CHARACTERIC FOR ERROR RETURNS
                   ;        NOTE:  WE PRESERVE ORIGINAL MANTISSA SIGN
                   ;        ON ENTRY D CONTAINS NEW CHARACTERISTIC TO BE STORED.
089F   2C          WCHAR:   INR     L             ;SET (H,L) TO POINT TO CHARA
08A0   2C                   INR     L             ;PART OF ABOVE
08A1   2C                   INR     L             ;PART OF ABOVE
08A2   7E                   MOV     A,M           ;LOAD CHARACTERISTIC A
                                                  ;AND MANTISSA SIGN
08A3   E6 80                ANI     200Q          ;JUST KEEP MANTISSA SIGN
08A5   B2                   ORA     D             ;OR IN NEW CHARACTERISTIC
08A6   77                   MOV     M,A           ;STORE IT BACK
08A7   C9                   RET                   ;RETURN WITH (H,L) POINT TO
                                                  ;OF RESULT
                                                  ;SOMEONE ELSE WILL FIX UP (H

                   ;  SUBROUTINE INDFC

                   ;     THIS ROUTINE WRITES A FLOATING INDEFINITE, SETS
                   ;     THIS WRITES WRITES A FLOATING POINT INDEFINITE
                   ;     AT (H,C), SETS THE CONDITION FLAG AND RETURNS

08A8   5D          INDFC:   MOV     E,L           ;SAVE LPTR IN E
08A9   69                   MOV     L,C           ;SET L=CPTR SO (H,L)=ADDR OF
08AA   CD 7F 0B             CALL    WIND          ;WRITE INDEFINITE
08AD   6B                   MOV     L,E           ;RESTORE L=LPTR
08AE   C9                   RET                   ;RETURN

                   ;  SUBROUTINE WZERC

                   ;     THIS ROUTINE WRITES A NORMAL FLOATING POINT ZERO
                   ;     AT (H,C), SETS THE CONDITION FLAG AND RETURNS

08AF   5D          WZERC:   MOV     E,L           ;SAVE LPTR IN E
08B0   69                   MOV     L,C           ;SETL=CPTR SO (H,L)=ADDR OF
08B1   CD 8D 0B             CALL    WZER          ;WRITE NORMAL ZERO
08B4   6B                   MOV     L,E           ;RESTORE L=LPTR
08B5   C9                   RET                   ;RETURN

                   ;  SUBROUTINE INCR

                   ;     THIS SUBROUTINE INCREMENTS THE CHARACTERISTIC
                   ;     OF THE FLOATING POINT NUMBER POINTED TO BY (H,L).
                   ;     WE TEST FOR OVERFLOW AND SET APPROPRIATE FLAG.
                   ;     (SEE ERROR RETURNS).

                   ;  REGISTERS ON EXIT:

                   ;     A = CONDITION FLAG (SEE ERROR RETURNS)
                   ;     D = CLOBBERED
                   ;     B,C,H,L = SAME AS ON ENTRY

08B6   CD 41 0B    INCR:    CALL    GCHAR         ;GET CHAR WITH SIGN EXTENDED
08B9   FE 3F                CPI     MAXCH         ;COMPARE WITH MAX CHAR PERMI
08BB   CA 76 0B             JZ      UFLW1         ;INCREMENT WOULD CAUSE OVERF
08BE   57                   MOV D,A               ;/SAVE IT IN D
08BF   14                   INR D                 ;/INCREMENT IT
08C0   C3 C6 0B             JMP     INCR2         ;JUMP AROUND ALTERNATE ENTRY
08C3   2C          INCR3:   INR     L             ;COME HERE TO STORE CHARACTE
08C4   2C                   INR     L             ;POINT (H,L) TO CHAR
08C5   2C                   INR     L             ;POINT (H,L) TO CHAR
08C6   3E 7F       INCR2:   MVI A,177Q
08C8   A2                   ANA D                 ;/KILL SIGN BIT
08C9   57                   MOV D,A               ;/BACK TO D
08CA   7E                   MOV A,M               ;/NOW SIGN IT
08CB   E6 80                ANI 200Q              ;/GET MANTISSA SIGN
08CD   B2                   ORA D                 ;/PUT TOGETHER
08CE   77                   MOV M,A               ;/STORE IT BACK
08CF   2D                   DCR L                 ;/NOW BACK TO BASE
08D0   2D                   DCR L                 ;/***TP
08D1   2D                   DCR L
08D2   AF          SCCFG:   XRA     A             ;SET SUCCESS FLAG
08D3   C9                   RET

                   ;  SUBROUTINE DECR

                   ;     THIS SUBROUTINE DECREMENTS THE CHARACTERISTIC
                   ;     OF THE FLOATING POINT NUMBER POINTED TO BY (H,L).
                   ;     WE TEST FOR UNDERFLOW AND SET APPROPRIATE FLAG.
                   ;     (SEE ERROR RETURNS).

                   ;  REGISTERS ON EXIT:

                   ;     A = CONDITION FLAG (SEE ERROR RETURNS)
                   ;     D = CLOBBERED
                   ;     B,C,H,L = SAME AS ON ENTRY

08D4   CD 41 0B    DECR:    CALL    GCHAR         ;GET CHAR WITH SIGN EXTENDED
08D7   FE C0                CPI     MINCH         ;COMPARE WITH MIN CHAR PERMI
08D9   CA 68 0B             JZ      UFLW1         ;DECREMENT WOULD CAUSE UNDER
08DC   57                   MOV     D,A           ;SAVE CHARACTERISTIC IN D
08DD   15                   DCR     D             ;DECREMENT CHARACTERISTIC
08DF   C3 C6 0B             JMP     INCR2         ;GO STORE IT BACK
                   ;
                   ;                              SUBROUTINE ADRS
                   ;                              RETURNS S?1 IF BASE 76
                   ;                              HAS A 1 IN MSB
08E1   5D          ADRS:    MOV E,L               ;SAVE BASE
08E2   69                   MOV L,C               ;BASE 76 TO L
08E3   7E                   MOV A,M               ;LOAD IT
08E4   B7                   ORA  A                ;SET FLAGS
08E5   6B                   MOV L,E               ;RESTORE BASE
08E6   C9                   RET
                   ;
                   ;                              SUBROUTINE TSTR
                   ;                              CHECKS C PTR TO SEE IF
                   ;                              NLSB?1
                   ;                              RETURNS Z?1 IF NOT
                   ;                              DESTROYS E,D
08E7   5D          TSTR:    MOV E,L               ;SAVE BASE
08E8   69                   MOV L,C               ;C PTR TO L
08E9   16 02                MVI D,2               ;MASK TO D
08EB   7E                   MOV A,M               ;LOAD VALUE
08EC   6B                   MOV L,E               ;RESTORE BASE
08ED   A2                   ANA D                 ;AND VALUE WITH MASK
08EE   C9                   RET
                   ;
                   ;                              SUBROUTINE ACPR
                   ;                              STORES A IN LOCATION OF CPTR
                   ;                              LPTR IN E
08EF   5D          ACPR:    MOV E,L               ;SAVE LPTR
08F0   69                   MOV L,C               ;CPTR TO L
08F1   77                   MOV M,A               ;STORE A
08F2   6B                   MOV L,E               ;RESTORE BASE
08F3   C9                   RET
                   ;
                   ;                              SUBROUTINE DCMP
                   ;                              COMPARES TWO DOUBLE LENGTH
                   ;                              WORDS
08F4   7E          DCMP:    MOV A,M               ;NUM MANTA TO A
08F5   5D                   MOV E,L               ;SAVE BASE IN E
08F6   68                   MOV L,B               ;BASE?3 TO L
08F7   BE                   CMP M                 ;COMPARE WITH DEN MANTA
08F8   6B                   MOV L,E               ;RETURN BASE TO L
08F9   C0                   RNZ                   ;RETURN IF NOT THE SAME
08FA   2C                   INR L                 ;L TO NUM MANTB
08FB   7E                   MOV A,M               ;LOAD IT
08FC   68                   MOV L,B               ;DEN MANTB ADD TO L
08FD   2C                   INR L                 ;BASE? 4 TO L
08FE   BE                   CMP M
08FF   6B                   MOV L,E
0C00   C0                   RNZ                   ;/***TP EXTENSION
0C01   2C                   INR L                 ;/NOW CHECK BYTE 3
0C02   2C                   INR L
0C03   7E                   MOV A,M               ;/GET FOR COMPARE
0C04   68                   MOV L,B
0C05   2C                   INR L
0C06   2C                   INR L                 ;/BYTE 3 NOW
0C07   BE                   CMP M                 ;/COMPARE
0C08   6B                   MOV L,E               ;/***TP - ALL DONE
0C09   C9                   RET
                   ;
                   ;                              SUBROUTINE DIVC
                   ;                              PERFORMS ONE CYCLE OF DOUBLE
                   ;                              PRECISION FLOATING PT DIVIDE
                   ;                              ENTER AT ENT1 ON FIRST CYCLE
                   ;                              ENTER AT ENT2 ALL THEREAFTER
0C0A   CD E9 0A    ENT2:    CALL DLST             ;SHIFT MOVING DIVIDEND
0C0D   DA 17 0C             JC   OVER             ;IF CARRY?1,NUM.GT.D
0C10   CD F4 0B    ENT1:    CALL DCMP             ;COMPARE NUM WITH DEN
0C13   D2 17 0C             JNC  OVER             ;IF CARRY NOT SET,NUM.GE.DEN
0C16   C9                   RET
0C17   CD 26 0B    OVER:    CALL DSUB             ;CALL DOUBLE SUBTRACT
0C1A   5D                   MOV E,L               ;SAVE BASE IN E
0C1B   69                   MOV L,C               ;BASE 26 TO L
0C1C   2C                   INR L                 ;BASE 27 TO L
0C1D   2C                   INR L                 ;/***TP
0C1E   7E                   MOV A,M
0C1F   C6 01                ADI 1                 ;ADD 1
0C21   77                   MOV M,A               ;PUT IT BACK
0C22   6B                   MOV L,E               ;RESTORE BASE TO L
0C23   C9                   RET
                   ;
                   ;                              SUBROUTINE LXFR
                   ;                              MOVES CPTR TO EPTR
                   ;                              MOVES 3 WORDS IF ENTER AT LXFR
0C24   16 04       LXFR:    MVI D,4               ;/MOVE 4 WORDS
0C26   69          REP5:    MOV L,C               ;CPTR TO L
0C27   7E                   MOV A,M               ;?CPTR> TO A
0C28   6B                   MOV L,E               ;EPTR TO L
0C29   77                   MOV M,A
0C2A   0C                   INR C                 ;/INCREMENT C
0C2B   1C                   INR E                 ;/INCREMENT E TO NEXT
0C2C   15                   DCR D                 ;/TEST FOR DONE
0C2D   C2 26 0C             JNZ REP5              ;/GO FOR FOR TILL D=0
0C30   7B                   MOV A,E               ;/NOW RESET C AND E
0C31   D6 04                SUI 4                 ;/RESET BACK BY 4
0C33   5F                   MOV E,A               ;/PUT BACK IN E
0C34   79                   MOV A,C               ;/NOW RESET C
0C35   D6 04                SUI 4                 ;/BY 4
0C37   4F                   MOV C,A               ;/BACK TO C
0C38   C9                   RET                   ;/DONE
                   ;
                   ;  SUBROUTINE LDCP
                   ;
                   ;     THIS SUBROUTINE COMPUTES THE CHARACTERISTIC
                   ;     FOR THE FLOATING DIVIDE ROUTINE
                   ;
                   ;  REGISTERS ON EXIT:
                   ;
                   ;     A = CONDITION FLAG (SEE ERROR RETURNS)
                   ;     D,E = GARBAGE
                   ;     B,C,H,L = SAME AS ON ENTRY
                   ;
                   ;  REGISTERS ON ENTRY:
                   ;
                   ;     (H,B) = ADDRESS OFF DIVISOR
                   ;     (H,C) = ADDRESS OF QUOTIENT
                   ;     (H,L) = ADDRESS OF DIVIDEND
                   ;
0C39   CD 4C 0B    LDCP:    CALL    CFCHE         ;SET E=CHAR(H,B), A=CHAR(H,L
0C3C   93                   SUB     E             ;SUBTRACT TO GET NEW CHARACT
0C3D   C3 44 0C             JMP     CCHK          ;GO CHECK FOR OVER/UNDERFLOW
                                                  ;AND STORE CHARACTERTISTIC
                   ;
                   ;  SUBROUTINE LMCP
                   ;
                   ;     THIS SUBROUTINE COMPUTES THE CHARACTERISTIC
                   ;     FOR THE FLOATING MULTIPLY ROUTINE.
                   ;
                   ;  REGISTERS ON EXIT:
                   ;
                   ;     A = CONDITION FLAG (SEE ERROR RETURNS)
                   ;     D,E = GARBAGE
                   ;     B,C,H,L = SAME AS ON ENTRY
                   ;
                   ;  REGISTERS ON ENTRY:
                   ;
                   ;     (H,B) = ADDRESS OFF MULTIPLICAND
                   ;     (H,C) = ADDRESS OF PRODUCT
                   ;     (H,L) = ADDRESS OF MULTIPLIER
                   ;
0C40   CD 4C 0B    LMCP:    CALL    CFCHE         ;SET E=CHAR(H,B), A=CHAR(H,L
0C43   83                   ADD     E             ;ADD TO GET NEW CHARACTERIST
                                                  ;NOW FALL INTO THE ROUTINE
                                                  ;WHICH CHECKS FOR OVER/UNDER
                                                  ;AND STORE CHARACTERTISTIC
                   ;
                   ;  SBUROUTINE CCHK
                   ;
                   ;     THIS SUBROUTINE CHECKS A CHARACTERISTIC IN
                   ;     THE ACCUMULATOR FOR OVERFLOW OR UNDERFLOW.
                   ;     IT THEN STORES THE CHARACTERISTIC, PRESERVING
                   ;     THE PREVIOUSLY COMPUTED MANTISSA SIGN.
                   ;
                   ;  REGISTERS ON ENTRY:
                   ;
                   ;     (H,L) = ADDRESS OF ONE OPERAND
                   ;     (H,B) = ADDRESS OF OTHER OPERAND
                   ;     (H,C) = ADDRESS OF RESULT
                   ;     A     = NEW CHARACTERISTIC OF  RESULT
                   ;
                   ;  REGISTERS ON EXIT:
                   ;
                   ;     A = CONDITION FLAG (SEE ERROR RETURNS)
                   ;     D,E = GARBAGE
                   ;     B,C,H,L = SAME AS ON ENTRY
                   ;
0C44               CCHK:                          ;ENTER HERE TO CHECK CHARACT
0C44   FE 40                CPI     100Q          ;CHECK FOR 0 TO +63
0C46   DA 53 0C             JC      STORC         ;JUMP IF OKAY
0C49   FE 80                CPI     200Q          ;CHECK FOR +64 TO +127
0C4B   DA 58 0C             JC      OFLWC         ;JUMP IF OVERFLOW
0C4E   FE C0                CPI     300Q          ;CHECK FOR -128 TO -65
0C50   DA 62 0C             JC      UFLWC         ;JUMP IF UNDERFLOW
0C53   5D          STORC:   MOV     E,L           ;SAVE L IN E
0C54   69                   MOV     L,C           ;LET L POINT TO RESULT
0C55   57                   MOV     D,A           ;SAVE CHARACTERISTIC IN D
0C56   CD C3 0B             CALL    INCR3         ;STORE CHARACTERISTIC
0C59   6B                   MOV     L,E           ;RESTORE L
0C5A   C9                   RET                   ;RETURN
                   ;
                   ;  SUBROUTINE OFLWC
                   ;
                   ;     THIS ROUTINE WRITES A FLOATING POINT OVERFLOW AT
                   ;     SETS THE CONDITION FLAG, AND RETURNS.
                   ;
0C5B   5D          OFLWC:   MOV     E,L           ;SAVE L IN E
0C5C   69                   MOV     L,C           ;SET L=CPTR, SO (H,L)=ADDR O
0C5D   CD 71 0B             CALL    WOVR          ;WRITE OUT OVERFLOW
0C60   6B                   MOV     L,E           ;RESTORE L
0C61   C9                   RET                   ;RETURN
                   ;
                   ;  SUBROUTINE UFLWC
                   ;
                   ;     THIS ROUTINE WRITES A FLOATING POINT UNDERFLOW AT
                   ;     SETS THE CONDITION FLAG, AND RETURNS.
                   ;
0C62   5D          UFLWC:   MOV     E,L           ;SAVE L IN E
0C63   69                   MOV     L,C           ;SET L=CPTR, SO (H,L)=ADDR O
0C64   CD 63 0B             CALL    WUND          ;WRITE OUT UNDEFLOW
0C67   6B                   MOV     L,E           ;RESTORE L
0C68   C9                   RET                   ;RETURN
                   ;
                   ;
                   ;  SUBROUTINE CSIGN
                   ;
                   ;     THIS SUBROUTINE COMPUTES AND STORE THE MANTISSA
                   ;     SIGN FOR THE FLOATING MULTIPLY AND DIVIDE ROUTINE
                   ;
                   ;  REGISTERS ON ENTRY:
                   ;
                   ;     (H,L) = ADDRESS OF ONE OPERAND
                   ;     (H,B) = ADDRESS OF OTHER OPERAND
                   ;     (H,C) = ADDRESS OF RESULT
                   ;
                   ;  REGISTERS ON EXIT:
                   ;
                   ;     A,D,E = GARBAGE
                   ;     B,C,H,L = SAME AS ON ENTRY
```

```
0C69  CD 79 0C   CSIGN:  CALL    MSFH         ;SET A=SIGN(H,L), E=SIGN(H,B
0C6C  AB                 XRA     E            ;EXCLUSIVE OR SIGNS TO GET N
0C6D  CD 71 0C           CALL    CSTR         ;STORE SIGN INTO RESULT
0C70  C9                 RET                  ;RETURN

                                 SUBROUTINE CSTR
                                 STORES VALUE IN A IN
                                 CPTR?2
                                 PUTS LPTR IN E
0C71  5D         CSTR:   MOV E,L          ;SAVE LPTR IN E
0C72  69                 MOV L,C          ;CPTR TO L
0C73  2C                 INR L            ;CPTR?2
0C74  2C                 INR L            ;TO L
0C75  2C                 INR L            ;/***TP
0C76  77                 MOV M,A          ;STORE ANSWER
0C77  6B                 MOV L,E          ;LPTR BACK TO L
0C78  C9                 RET

         SUBROUTINE MSFH

             THIS SUBROUTINE FETCHES THE SIGNS OF THE MANTISSA
             OF THE FLOATING POINT NUMBERS POINTED TO BY (H,L)
             AND (H,B) INTO THE A AND E REGISTERS RESPECTIVELY

         REGISTERS ON EXIT:

             A = SIGN OF MANTISSA OF (H,L)
             E = SIGN OF MANTISSA OF (H,B)
             B,C,D,H,L = SAME AS ON ENTRY
0C79  5D         MSFH:   MOV E,L          ;SAVE LPTR
0C7A  68                 MOV L,B          ;BPTR TO L
0C7B  2C                 INR L            ;BPTR?2
0C7C  2C                 INR L            ;/***TP
0C7D  2C                 INR L            ;TO L
0C7E  7E                 MOV A,M          ;?BPTR?2>TO A
0C7F  E6 80              ANI 128          ;SAVE MANT SIGN
0C81  6B                 MOV L,E          ;LPTR BACK TO L
0C82  5F                 MOV E,A          ;STORE BPTR MANT SIGN
0C83  2C                 INR L            ;LPTR?2
0C84  2C                 INR L            ;/***TP
0C85  2C                 INR L            ;TO L
0C86  7E                 MOV A,M          ;?LPTR?2>TO A
0C87  E6 80              ANI 128          ;SAVE LPTR MANT SIGN
0C89  2D                 DCR L            ;LPTR BACK
0C8A  2D                 DCR L            ;TO L
0C8B  2D                 DCR L            ;/***TP
0C8C  C9                 RET
                                 SUBROUTINE BCTL
                                 MOVES BPTR CHAR TO LPTR CHAR
                                 DESTROYS E
0C8D  5D         BCTL:   MOV E,L          ;LPTR TO E
0C8E  68                 MOV L,B          ;BPTR TO L
0C8F  2C                 INR L            ;BPTR ?2
0C90  2C                 INR L            ;/***TP
0C91  2C                 INR L            ;TO L
0C92  7E                 MOV A,M          ;BPTR CHAR TO A
0C93  6B                 MOV L,E          ;LPTR TO L
0C94  2C                 INR L            ;LPTR ?2
0C95  2C                 INR L            ;TO L
0C96  2C                 INR L            ;/***TP
0C97  77                 MOV M,A          ;STORE BPTR CHAR IN LPTR CHAR
0C98  6B                 MOV L,E          ;LPTR TO L
0C99  C9                 RET

;**********************************************************
;        //// 5 DIGIT FLOATING PT. OUTPUT
;**********************************************************

         *******ROUTINE TO CONVERT FLOATING PT.
         ***NUMBERS TO ASCII AND OUTPUT THEM VIA A SUBROU
         ***CALLED OUTR  -  NOTE: THIS IS CURRENTLY SET
         ***TO DDT'S OUTPUT ROUTINE

0C9A  CD 0A 0A   CVRT:   CALL    ZCHK         ;CHECK FOR NEW ZERO
0C9D  C2 B2 0C           JNZ     NNZRO        ;NOT ZERO
0CA0  0C                 INR     C            ;IT WAS, OFFSET C BY 2
0CA1  0C                 INR     C
0CA2  69                 MOV     L,C
0CA3  CD 80 0B           CALL    WZER         ;WRITE ZERO
0CA6  2C                 INR     L            ;PNT TO DECIMAL EXPONEN
0CA7  2C                 INR     L
0CA8  2C                 INR     L
0CA9  2C                 INR     L
0CAA  AF                 XRA     A            ;SET IT TO ZERO
0CAB  77                 MOV     M,A
0CAC  CD 93 0D           CALL    SIGN         ;WRITE OUT SIGN
0CAF  C3 11 0D           JMP     MDSKP        ;OUTPUT IT
0CB2  56         NNZRO:  MOV     D,M          ;/GET THE NUMBER TO CON
0CB3  2C                 INR L
0CB4  46                 MOV B,M
0CB5  2C                 INR L
0CB6  5E                 MOV E,M
0CB7  2C                 INR L            ;/4 WORD***TP
0CB8  7E                 MOV A,M          ;/***TP
0CB9  0C                 INR C            ;/OFFSET SCRATCH POINTER BY 2
0CBA  0C                 INR C
0CBB  69                 MOV L,C          ;/L NOT NEEDED ANY MORE
0CBC  72                 MOV M,D          ;/SAVE NUMBER IN SCRATCH
0CBD  2C                 INR L
0CBE  70                 MOV M,B
0CBF  2C                 INR L
0CC0  73                 MOV M,E          ;/***TP
0CC1  2C                 INR L            ;/***TP
0CC2  47                 MOV B,A          ;/SAVE COPY OF CHAR & SIGN
0CC3  E6 7F              ANI 177Q         ;GET ONLY CHAR.
0CC5  77                 MOV M,A          ;/SAVE ABS(NUMBER)
0CC6  FE 40              CPI 100Q         ;CK FOR ZERO
0CC8  CA CF 0C           JZ   NZRO
0CCB  D6 01              SUI 1            ;/GET SIGN OF DEC. EXP
0CCD  E6 40              ANI 100Q         ;/GET SIGN OF CHAR.
0CCF  07         NZRO:   RLC              ;MOVE IT TO SIGN POSITION
0CD0  2C                 INR L            ;/MOVE TO DECIMAL EXP.
0CD1  77                 MOV M,A          ;/SAVE SIGN OF EXP.
0CD2  78                 MOV A,B          ;/GET MANT. SIGN BACK
0CD3  CD 93 0D           CALL SIGN        ;/OUTPUT SIGN
0CD6  2E 17              MVI L,(TEN5 AND 377Q)   ;/TRY MULT. OR DIV. BY 10
0CD8  CD F4 0D           CALL COPT        ;/MAKE A COPY IN RAM
0CDB  CD A0 0D   TSTR:   CALL GCHR        ;/GET CHAR. OF NUMBER
0CDE  47                 MOV B,A          ;/SAVE A COPY
0CDF  E6 40              ANI 100Q         ;/GET ABSOLUTE VALUE OF CHAR
0CE1  78                 MOV A,B          ;/INCASE PLUS
0CE2  CA E8 0C           JZ   GOTV        ;/ALREADY PLUS
0CE5  3E 80              MVI A,200Q       ;/MAKE MINUS INTO PLUS
0CE7  90                 SUB B            ;/PLUS=200B-CHAR
0CE8  FE 12      GOTV:   CPI 22Q          ;/TEST FOR USE OF 100000
0CEA  FA F6 0C           JM   TRY1        ;/WONT GO
0CED  CD A6 0D           CALL MORD        ;/WILL GO SO DO IT
0CF0  C6 05              ADI 5            ;/INCREMENT DEC. EXPONENT BY 5
0CF2  77                 MOV M,A          ;/UPDATE MEM
0CF3  C3 DB 0C           JMP TSTR         ;/GO TRY AGAIN
0CF6  2E 1B      TRY1:   MVI L,(TEN AND 377Q)    ;/NOW USE JUST TEN
0CF8  CD F4 0D           CALL COPT        ;/PUT IT IN RAM
0CFB  CD A0 0D   TST1:   CALL GCHR        ;/GET CHARACTERISTIC
0CFE  FE 01              CPI 1            ;/MUST GET IN RANGE 1 TO 6
0D00  F2 0C 0D           JP   OK1         ;/ATLEAST ITS 1 OR BIGGER
0D03  CD A6 0D   MDGN:   CALL MORD        ;/MUST MUL OF DIV BY 10
0D06  C6 01              ADI 1            ;/INCREMENT DECIMAL EXP.
0D08  77                 MOV M,A          ;/UPDATE MEM
0D09  C3 FB 0C           JMP TST1         ;/NOW TRY AGAIN
0D0C  FE 07      OK1:    CPI 7            ;/TEST FOR LESS THAN 7
0D0E  F2 03 0D           JP   MDGN        ;/NOPE - 7 OR GREATER
0D11  69         MDSKP:  MOV    L,C           ;/SET UP DIGIT COUNT
0D12  2D                 DCR L
0D13  2D                 DCR L            ;/IN 1ST WORD OF SCRATCH
0D14  36 05              MVI M,5          ;/5 DIGITS
0D16  5F                 MOV E,A          ;/SAVE CHAR. AS LEFT SHIFT COU
0D17  CD 79 0D           CALL LSFT        ;/SHIFT LEFT PROPER NUMBER
0D1A  FE 0A              CPI 12Q          ;/TEST FOR 2 DIGITS HERE
0D1C  F2 CC 0D           JP   TWOD        ;/JMP IF 2 DIGITS TO OUTPUT
0D1F  CD 3D 0D           CALL DIGO        ;/OUTPUT FIRST DIGIT
0D22  CD 51 0D   POPD:   CALL MULTT       ;/MULTIPLY THE NUMBER BY 10
0D25  CD 3D 0D   INPOP:  CALL DIGO        ;/PRINT DIGIT IN A
0D28  C2 22 0D           JNZ POPD         ;/MORE DIGITS?
0D2B  3E C5              MVI A,305Q       ;/NO SO PRINT E
0D2D  CD FD 0F           CALL OUTR        ;/BASIC CALL TO OUTPUT
0D30  CD C1 0D           CALL GETEX       ;/GET DECIMAL EXP
0D33  47                 MOV B,A          ;/SAVE A COPY
0D34  CD 93 0D           CALL SIGN        ;/OUTPUT SIGN
0D37  78                 MOV A,B          ;/GET EXP BACK
0D38  E6 3F              ANI 77Q          ;/GET GOOD BITS
0D3A  CD E3 0D           CALL CTWO        ;/GO CONVERT 2 DIGITS
0D3D  C6 B0      DIGO:   ADI 260Q         ;/MAKE A INTO ASCII
0D3F  CD FD 0F           CALL OUTR        ;/OUTPUT DIGIT
0D42  69                 MOV L,C          ;/GET DIGIT COUNT
0D43  2D                 DCR L            ;/BACK UP TO DIGIT COUNT
0D44  2D                 DCR L
0D45  7E                 MOV A,M          ;/TEST FOR DECIMAL PT
0D46  FE 05              CPI 5            ;/PRINT . AFTER 1ST DIGIT
0D48  3E AE              MVI A,256Q       ;/JUST IN CASE
0D4A  CC FD 0F           CZ   OUTR        ;/OUTPUT . IF 1ST DIGIT
0D4D  56                 MOV D,M          ;/NOW DECREMENT DIGIT COUNT
0D4E  15                 DCR D
0D4F  72                 MOV M,D          ;/UPDATE MEM AND LEAVE FLOPS S
0D50  C9                 RET              ;/SERVES AS TERM FOR DIGO & CV
0D51  1E 01      MULTT:  MVI E,1          ;/MULT. BY 10 (START WITH X2)
0D53  CD 79 0D           CALL LSFT        ;/LEFT SHIFT 1 = X2
0D56  69                 MOV L,C          ;/SAVE X2 IN "RESULT"
0D57  2D                 DCR L            ;/SET TO TOP OF NUMBER
0D58  79                 MOV A,C          ;/SET C TO RESULT
0D59  C6 09              ADI 11Q
0D5B  4F                 MOV C,A          ;/NOW C SET RIGHT
0D5C  7C                 MOV A,H          ;/SHOW RAM TO RAM TRANSFER
0D5D  CD 02 0E           CALL COPY        ;/SAVE X2 FINALLY
0D60  79                 MOV A,C          ;/MUST RESET C
0D61  D6 09              SUI 11Q          ;/BACK TO NORMAL
0D63  4F                 MOV C,A
0D64  1E 02              MVI E,2          ;/NOW GET (X2)X4=X8
0D66  69                 MOV L,C          ;/BUT MUST SAVE OVERFLOW
0D67  2D                 DCR L
0D68  CD 7D 0D           CALL TLP2        ;/GET X8
0D6B  69                 MOV L,C          ;/SET UP TO CALL DADD
0D6C  79                 MOV A,C          ;/SET B TO X2
0D6D  C6 0A              ADI 12Q          ;/TO X2
0D6F  47                 MOV B,A
0D70  CD 06 0B           CALL DADD        ;/ADD TWO LOW WORDS
0D73  2D                 DCR L            ;/BACK UP TO OVERFLOW
0D74  7E                 MOV A,M          ;/GET IT
0D75  68                 MOV L,B          ;/NOW SET TO X2 OVERFLOW
0D76  2D                 DCR L            ;/ITS AT B-1
0D77  8E                 ADC M            ;/ADD WITH CARRY - CARRY WAS P
0D78  C9                 RET              ;/ALL DONE, RETURN OVERFLOW IN
0D79  69         LSFT:   MOV L,C          ;/SET PTR FOR LEFT SHIFT OF NU
0D7A  2D                 DCR L            ;/BACK UP TO OVERFLOW
0D7B  AF                 XRA A            ;/OVERFLOW=0 1ST TIME
0D7C  77         TLOOP:  MOV M,A          ;/SAVE OVERFLOW
0D7D  1D         TLP2:   DCR E            ;/TEST FOR DONE
0D7E  F8                 RM               ;/DONE WHEN E MINUS
0D7F  2C                 INR L            ;/MOVE TO LOW
0D80  2C                 INR L
0D81  2C                 INR L            ;/***TP EXTENSION
0D82  7E                 MOV A,M          ;/SHIFT LEFT 4 BYTES
0D83  17                 RAL
0D84  77                 MOV M,A          ;/PUT BACK
0D85  2D                 DCR L            ;/***TP - ALL DONE
0D86  7E                 MOV A,M          ;/GET LOW
0D87  17                 RAL              ;/SHIFT LEFT 1
0D88  77                 MOV M,A          ;/RESTORE IT
0D89  2D                 DCR L            ;/BACK UP TO HIGH
0D8A  7E                 MOV A,M          ;/GET HIGH
0D8B  17                 RAL              ;/SHIFT IT LEFT WITH CARRY
0D8C  77                 MOV M,A          ;/PUT IT BACK
0D8D  2D                 DCR L            ;/BACK UP TO OVERFLOW
0D8E  7E                 MOV A,M          ;/GET OVERFLOW
0D8F  17                 RAL              ;/SHIFT IT LEFT
0D90  C3 7C 0D           JMP TLOOP        ;/GO FOR MORE
0D93  E6 80      SIGN:   ANI 200Q         ;/GET SIGN BIT
0D95  3E A0              MVI A,240Q       ;/SPACE INSTEAD OF PLUS
0D97  CA 9C 0D           JZ   PLSV        ;/TEST FOR +
0D9A  3E AD              MVI A,255Q       ;/NEGATIVE
0D9C  CD FD 0F   PLSV:   CALL OUTR        ;/OUTPUT SIGN
0D9F  C9                 RET
0DA0  69         GCHR:   MOV L,C          ;/GET CHARCTERISTIC
0DA1  2C         GETA:   INR L            ;/MOVE TO IT
0DA2  2C                 INR L
0DA3  2C                 INR L            ;/***TP
0DA4  7E                 MOV A,M          ;/FETCH INTO A
0DA5  C9                 RET              ;/DONE
0DA6  CD C1 0D   MORD:   CALL GETEX       ;/MUL OR DIV DEPENDING ON EXP
0DA9  5F                 MOV E,A          ;/SAVE DECIMAL EXP
0DAA  45                 MOV B,L          ;/SET UP TO MULT OR DIV
0DAB  04                 INR B            ;/NOW BOP POINTER SET
0DAC  69                 MOV L,C          ;/L POINTS TO NUMBER TO CONVERT
0DAD  79                 MOV A,C          ;/POINT C AT "RESULT" AREA
0DAE  C6 09              ADI 11Q          ;/IN SCRATCH
0DB0  4F                 MOV C,A          ;/NOW C SET RIGHT
0DB1  7B                 MOV A,E          ;/NOW TEST FOR MUL
0DB2  E6 80              ANI 200Q         ;/TEST NEGATIVE DEC. EXP.
0DB4  CA C6 0D           JZ   DIVIT       ;/IF EXP IS + THEN DIVIDE
0DB7  CD 5F 0A           CALL LMUL        ;/MULT.
0DBA  79         FINUP:  MOV A,C          ;/SAVE LOC. OF RESULT
0DBB  4D                 MOV C,L          ;/C=LOC OF NUMBER (IT WAS DEST
0DBC  6F                 MOV L,A          ;/SET L TO LOC. OF RESULT
0DBD  7C                 MOV A,H          ;/SHOW RAM TO RAM TRANSFER
0DBE  CD 02 0E           CALL COPY        ;/MOVE RESULT TO NUMBER
0DC1  69         GETEX:  MOV L,C          ;/NOW GET DECIMAL EXP
0DC2  2C                 INR L
0DC3  C3 A1 0D           JMP GETA         ;/USE PART OF GCHR
0DC6  CD 00 09   DIVIT:  CALL LDIV        ;/DIVIDE
0DC9  C3 BA 0D           JMP FINUP
0DCC  CD E3 0D   TWOD:   CALL CTWO        ;/CONVERT TO 2 DIGITS
0DCF  47                 MOV B,A          ;/SAVE ONES DIGIT
0DD0  CD C1 0D           CALL GETEX       ;/GET DECIMAL EXP
0DD3  5F                 MOV E,A          ;/SAVE A COPY
0DD4  E6 80              ANI 200Q         ;/TEST FOR NEGATIVE
0DD6  CA DF 0D           JZ   ADD1        ;/BUMP EXP BY 1 SINCE 2 DIGITS
0DD9  1D                 DCR E            ;/DECREMENT NEGATIVE EXP SINCE
0DDA  73         FINIT:  MOV M,E          ;/RESTORE EXP WITH NEW VALUE
0DDB  78                 MOV A,B          ;/NOW DO 2ND DIGIT
0DDC  C3 25 0D           JMP INPOP        ;/GO OUT 2ND AND REST FO DIGIT
0DDF  1C         ADD1:   INR E            ;/COMPENSATE FOR 2 DIGITS
0DE0  C3 DA 0D           JMP FINIT
0DE3  1E FF      CTWO:   MVI E,377Q       ;/CONVERT 2 DIGIT BIN TO BCD
0DE5  1C         LOOP:   INR E            ;/ADD UP TENS DIGIT
0DE6  D6 0A              SUI 12Q          ;/SUBTRACT 10
0DE8  F2 E5 0D           JP   LOOP        ;/TIIL NEGATIVE RESULT
0DEB  C6 0A              ADI 12Q          ;/RESTORE ONES DIGIT
0DED  47                 MOV B,A          ;/SAVE ONES DIGIT
0DEE  7B                 MOV A,E          ;/GET TENS DIGIT
0DEF  CD 3D 0D           CALL DIGO        ;/OUTPUT IT
0DF2  78                 MOV A,B          ;/SET A TO 2ND DIGIT
0DF3  C9                 RET
0DF4  79         COPT:   MOV A,C          ;/COPY FROM 10?N TO RAM
0DF5  C6 05              ADI 5
0DF7  4F                 MOV C,A          ;/SET C TO PLACE TO PUT
0DF8  3E 0E              MVI A,(TEN5/256)
0DFA  CD 02 0E           CALL COPY        ;/COPY IT
0DFD  79                 MOV A,C          ;/NOW RESET C
0DFE  D6 05              SUI 5
0E00  4F                 MOV C,A          ;/ITS RESET
0E01  C9                 RET
0E02  44         COPY:   MOV B,H          ;/SAVE RAM H
0E03  67                 MOV H,A          ;/SET TO SOURCE H
0E04  7E                 MOV A,M          ;/GET 4 WORDS INTO THE REGS.
0E05  2C                 INR L
0E06  56                 MOV D,M
0E07  2C                 INR L
0E08  5E                 MOV E,M
0E09  2C                 INR L
0E0A  6E                 MOV L,M          ;/LAST ONE ERASES L
0E0B  60                 MOV H,B          ;/SET TO DESTINATION RAM
0E0C  45                 MOV B,L          ;/SAVE 4TH WORD IN B
0E0D  69                 MOV L,C          ;/SET TO DESTINATION
0E0E  77                 MOV M,A          ;/SAVE FIRST WORD
0E0F  2C                 INR L
0E10  7E                 MOV A,M          ;/SAVE THIS WORD IN A (INPUT S
0E11  72                 MOV M,D          ;/NOW PUT 2ND WORD
0E12  2C                 INR L
0E13  73                 MOV M,E
0E14  2C                 INR L
0E15  70                 MOV M,B          ;/ALL 4 COPIED NOW
0E16  C9                 RET              ;/ALL DONE

0E17  C3 50 00 11  TEN5:  DB 303Q,120Q,00,21Q   ;/303240(8) = 100000.
0E1B  A0 00 00 04  TEN:   DB 240Q,0Q,0Q,4Q      ;/12(8) = 10

                   SCRATCH MAP FOR I/O CONVERSION ROUTINES

                   RELATIVE TO (C+2)USE
                   C-2          DIGIT COUNT
                   C-1          OVERFLOW
                   C            HIGH NUMBER - MANTISSA
                   C+1          LOW NUMBER
                   C+2          CHARACTERISTIC
                   C+3          DECIMAL EXPONEXT (SIGN & MAG.)
                   C+4          TEN**N
                   C+5          TEN**N
                   C+6          TEN**N
                   C+7          RESULT OF MULT & DIV
                   C+8          AND TEMP FOR X2
                   C+9          "        "
                   C+10         L FOR NUMBER TO GO INTO (INPUT O
                   C+11         DIGIT JUST INPUT (INPUT ONLY)

                                /*****BEGIN INPUT**************

0E1F  37         ERR:    STC              ;ERROR FLAG
0E20  C9                 RET              ;AND RETURN

;**********************************************************
;        //// 4 1/2 DIGIT INPUT ROUTINE
;**********************************************************

                                /L POINTS TO WHERE TO PUT INPUT
                                /C POINTS TO 13(10) WORDS OF SCR
0E21  45         INPUT:  MOV B,L          ;/SAVE ADDRESS WHERE DATA IS T
0E22  79                 MOV A,C          ;/IN SCRATCH
0E23  C6 0F              ADI 17Q          ;/COMPUTE LOC. IN SCRATCH
0E25  6F                 MOV L,A
0E26  70                 MOV M,B          ;/PUT IT
0E27  0C                 INR C            ;/OFFSET SCRATCH POINTER
```

```
0E28  0C              INR C                  ;/BY 2
0E29  CD 30 0F        CALL ZROIT             ;/ZERO NUMBER
0E2C  2C              INR L                  ;/AND ZERO
0E2D  77              MOV M,A                ;/DECIMAL EXPONENT
0E2E  CD CC 0E        CALL GNUM              ;/GET INTEGER PART OF NUM
0E31  FE FF           CPI 376Q               ;/TERM=.?
0E33  CA 86 0E        JZ   DECPT             ;/YES
0E36  FE 15   TSTEX:  CPI 25Q                ;/TEST FOR E
0E38  CA 98 0E        JZ   INEXP             ;/YES - HANDLE EXP
0E3B  FE F0           CPI 360Q               ;/TEST FOR SPACE TERM (2408-26
0E3D  C2 1F 0E        JNZ ERR                ;/NOT LEGAL TERM
0E40  CD 15 0F        CALL FLTSGN            ;/FLOAT # AND SIGN IT
0E43  CD C1 00 SCALF: CALL GETEX             ;/GET DECIMAL EXP
0E46  E6 7F           ANI 177Q               ;/GET GOOD BITS
0E48  5F              MOV E,A                ;/SAVE COPY
0E49  E6 40           ANI 100Q               ;/GET SIGN OF EXP
0E4B  07              RLC                    ;/INTO SIGN BIT
0E4C  B7              ORA A                  ;/SET FLOPS
0E4D  47              MOV B,A                ;/SAVE SIGN
0E4E  7B              MOV A,E                ;/GET EXP BACK
0E4F  CA 55 0F        JZ   APLS              ;/JMP IS +
0E52  3E 80           MVI A,200Q             ;/MAKE MINUS +
0E54  93              SUB E                  ;/NOW ITS +
0E55  80      APLS:   ADD B                  ;/SIGN NUMBER
0E56  77              MOV M,A                ;/SAVE EXP (SIGN & MAG.)
0E57  2E 17           MVI L,(TEN5 AND 377Q)  ;/TRY MORD WITH 10**5 FIR
0E59  CD F4 00        CALL CCPT              ;/TRANSFER TO RAM
0E5C  CD C1 00        CALL GETEX             ;/GET DECIMAL EXP
0E5F  E6 3F   INT5:   ANI 77Q                ;/GET MAG. OF EXP
0E61  FE 05           CPI 5Q                 ;/TEST FOR USE OF 10**5
0E63  FA 6F 0E        JM   TRYTN             ;/WONT GO - TRY 10
0E66  CD A6 00        CALL MCPD              ;/WILL GO SO DO IT
0E69  D6 05           SUI 5Q                 ;/MAG = MAG -5
0E6B  77              MOV M,A                ;/UPDATE DEC. EXP IN MEM
0E6C  C3 5F 0E        JMP INT5               ;/GO TRY AGAIN
0E6F  2E 18   TRYTN:  MVI L,(TEN AND 377Q)   ;/PUT TEN IN RAM
0E71  CD F4 00        CALL CCPT
0E74  CD C1 00        CALL GETEX             ;/SET UP FOR LOOP
0E77  E6 3F   INT1:   ANI 77Q                ;/GET MAGNITUDE
0E79  B7              ORA A                  ;/TEST FOR 0
0E7A  CA 19 0F        JZ   SAVEN             ;/DONE, MOVE NUM OUT AND GET C
0E7D  CD A6 0D        CALL MCRD              ;/NOT DONE - DO 10
0E80  D6 01           SUI 1Q                 ;/EXP = EXP -1
0E82  77              MOV M,A                ;/UPDATE MEM
0E83  C3 77 0E        JMP INT1               ;/TRY AGAIN
0E86  69      DECPT:  MOV L,C                ;/ZERO DIGIT COUNT
0E87  2D              DCR L                  ;/SINCE ITS NECESSARY
0E88  2D              DCR L                  ;/TO COMPUTE EXP.
0E89  36 00           MVI M,0                ;/ZEROED
0E8B  CD 0F 0F        CALL EP1               ;/GNUM IN MIDDLE
0E8E  5F              MOV E,A                ;/SAVE TERMINATOR
0E8F  69              MOV L,C                ;/MOVE DIGIT COUNT TO EXP
0E90  2D              DCR L                  ;/BACK UP TO DIGIT COUNT
0E91  2D              DCR L
0E92  46              MOV B,M                ;/GOT DIGIT COUNT
0E93  CD C1 0D        CALL GETEX             ;/SET L TO DEC. EXP
0E96  70              MOV M,B                ;/PUT EXP
0E97  78              MOV A,E                ;/TERM BACK TO A
0E98  C3 36 0E        JMP TSTEX              ;/TEST FOR E+OR-XX
0E9B  CD 15 0F INEXP: CALL FLTSGN            ;/FLOAT AND SIGN NUMBER
0E9E  CD 19 0F        CALL SAVEN             ;/SAVE NUMBER IN (L) TEMP
0EA1  CD 30 0F        CALL ZROIT             ;/ZERO OUT NUM. FOR INPUTTING
0EA4  CD CC 0E        CALL GNUM              ;/NOW INPUT EXPONENT
0EA7  FE F0           CPI 360Q               ;/TEST FOR SPACE TERM.
0EA9  C2 1F 0E        JNZ ERR                ;/NOT LEGAL - TRY AGAIN
0EAC  69              MOV L,C                ;/GET EXP OUT OF MEM
0EAD  2C              INR L                  ;/***TP
0EAE  2C              INR L                  ;/EXP LIMITED TO 5 BITS
0EAF  7E              MOV A,M                ;/GET LOWEST 8 BITS
0EB0  E6 1F           ANI 37Q                ;/GET GOOD BITS
0EB2  47              MOV B,A                ;/SAVE THEM
0EB3  2C              INR L                  ;/GET SIGN OF EXP
0EB4  7E              MOV A,M                ;/INTO A
0EB5  B7              ORA A                  ;/SET FLOPS
0EB6  78              MOV A,B                ;/INCASE NOTHING TO DO
0EB7  FA BD 0E        JM   USEIT             ;/IF NEG. USE AS +
0EBA  3E 00           MVI A,0Q               ;/IF + MAKE -
0EBC  90              SUB B                  ;/0-X = -X
0EBD  2C      USEIT:  INR L                  ;/POINT AT EXP
0EBE  86              ADD M                  ;/GET REAL DEC. EXP
0EBF  77              MOV M,A                ;/PUT IN MEM
0EC0  79              MOV A,C                ;/NOW GET NUMBER BACK
0EC1  C6 0D           ADI 15Q                ;/GET ADD OF L
0EC3  6F              MOV L,A                ;/L POINTS TO L OF NUMBER
0EC4  6E              MOV L,M                ;/NOW L POINTS TO NUMBER
0EC5  7C              MOV A,H                ;/RAM TO RAM COPY
0EC6  CD 02 0E        CALL COPY              ;/COPY IT BACK
0EC9  C3 43 0E        JMP SCALE              ;/NOW ADJUST FOR EXP
0ECC  CD FA 0F GNUM:  CALL INP               ;/GET A CHAR
0ECF  FE A0           CPI 240Q               ;/IGNORE LEADING SPACES
0ED1  CA CC 0E        JZ   GNUM
```



```
0ED4  FE AD           CPI 255Q               ;/TEST FOR -
0ED6  C2 E2 0F        JNZ TRYP               ;/NOT MINUS
0ED9  69              MOV L,C                ;/MINUS SO SET SIGN
0EDA  2C              INR L                  ;/IN CHAR LOC.
0EDB  2C              INR L                  ;/***TP
0EDC  2C              INR L
0EDD  36 80           MVI M,200Q             ;/SET - SIGN
0EDF  C3 CC 0E        JMP GNUM
0EE2  FE AB   TRYP:   CPI 253Q               ;/IGNORE +
0EE4  CA CC 0E        JZ   GNUM
0EE7  D6 B0   TSTN:   SUI 260Q               ;/STRIP ASCII
0EE9  F8              RM                     ;/RETURN IF TERM
0EEA  FE 0A           CPI 12Q                ;/TEST FOR NUMBER
0EEC  F0              RP                     ;/ILLEGAL
0EED  5F              MOV E,A                ;/SAVE DIGIT
0EEE  CD 2A 0F        CALL GETN              ;/LOC. OF DIGIT STORAGE TO L
0EF1  73              MOV M,E                ;/SAVE DIGIT
0EF2  CD 51 0D        CALL MULTT             ;/MULT NUMBER BY 10
0EF5  B7              ORA A                  ;/TEST FOR TOO MANY DIGITS
0EF6  C0              RNZ                    ;/TOO MANY DIGITS
0EF7  CD 2A 0F        CALL GETN              ;/GET DIGIT
0EFA  69              MOV L,C                ;/SET L TO NUMBER
0EFB  2C              INR L
0EFC  2C              INR L                  ;/***TP
0EFD  86              ADD M                  ;/ADD IN THE DIGIT
0EFE  77              MOV M,A                ;/PUT RESULT BACK
0EFF  2D              DCR L                  ;/NOW DO HIGH
0F00  7E              MOV A,M                ;/GET HIGH TO ADD IN CARRY
0F01  CE 00           ACI 0Q                 ;/ADD IN CARRY
0F03  77              MOV M,A                ;/UPDATE HIGH
0F04  2D              DCR L                  ;/***TP EXTENSION
0F05  7E              MOV A,M
0F06  CE 00           ACI 0Q                 ;/ADD IN CARRY
0F08  77              MOV M,A                ;/***TP ALL DONE
0F09  D8              RC                     ;/OVERFLOW ERROR
0F0A  2D              DCR L                  ;/BUMP DIGIT COUNT NOW
0F0B  2D              DCR L
0F0C  46              MOV B,M                ;/GET DIGIT COUNT
0F0D  04              INR B                  ;/BUMP DIGIT COUNT
0F0E  70              MOV M,B                ;/UPDATE DIGIT COUNT
0F0F  CD FA 0F EP1:   CALL INP               ;/GET NEXT CHAR
0F12  C3 E7 0E        JMP TSTN               ;/MUST BE NUM. OR TERM
0F15  69      FLTSGN: MOV L,C                ;POINT L AT NUMBER TO FLOAT
0F16  C3 05 0A        JMP FLOAT              ;GO FLOAT IT
0F19  79      SAVEN:  MOV A,C                ;/PUT NUMBER IN (L)
0F1A  C6 0D           ADI 15Q                ;/GET ADD OF L
0F1C  6F              MOV L,A
0F1D  5E              MOV E,M                ;/GET L OF RESULT
0F1E  6B              MOV L,E                ;/POINT L AT (L)
0F1F  2C              INR L                  ;/SET TO 2ND WORD TO SAVE C
0F20  71              MOV M,C                ;/SAVE C IN (L) +1 SINCE IT WI
0F21  69              MOV L,C                ;/SET UP TO CALL COPY
0F22  4B              MOV C,E                ;/NOW L&C SET
0F23  7C              MOV A,H                ;/RAM TO RAM COPY
0F24  CD 02 0F        CALL COPY              ;/COPY TO L
0F27  4F              MOV C,A                ;/(L)+1 RETURNED HERE SO SET A
0F28  B7              ORA  A                 ;MAKE SURE CY=0 (NO ERR
0F29  C9              RET                    ;/NOW EVERYTHING HUNKY-DORRY
0F2A  79      GETN:   MOV A,C                ;/GET DIGIT
0F2B  C6 0E           ADI 16Q                ;/LAST LOC. IN SCRATCH
0F2D  6F              MOV L,A                ;/PUT IN L
0F2E  7E              MOV A,M                ;/GET DIGIT
0F2F  C9              RET
0F30  69      ZROIT:  MOV L,C                ;/ZERO NUMBER
0F31  AF              XRA A
0F32  77              MOV M,A                ;/***TP
0F33  2C              INR L                  ;/***TP
0F34  77              MOV M,A
0F35  2C              INR L
0F36  77              MOV M,A
0F37  2C              INR L                  ;/NOW SET SIGN TO +
0F38  77              MOV M,A
0F39  C9              RET                    ;/DONE
00DB          READ    EQU    333Q            ;ODT READ ROUTINE
              ; CONTAIN LOW BYTE OF TWO BYTE VALUE. RETURNS CY=1 IF
              ; BC>DE, CY=0 IF BC<DE: Z=1 IF BC=DE.
0F3A  7B      DCOMP:  MOV    A,E
0F3B  B9              CMP    C
0F3C  C0              RNZ
0F3D  7A              MOV    A,D
0F3E  B8              CMP    B
0F3F  C9              RET
              ; ROUTINE TO INPUT CHAR FROM TTY
0F40  C5      CHAR2:  PUSH   B
0F41  CD DB 00        CALL   READ            ;INPUT FROM ODT
0F44  78              MOV    A,B             ;GET CHAR. TO A REG.
0F45  C1              POP    B               ;RESTORE B,C
0F46  C9              RET
              ; ROUTINE TO ADJUST VALUES OF BIN, FORWARD PNT. AND
              ; LINE LENGTH OF SOURCE LINE. PASSED ADD OF TEMP VARIABL
              ; CONTAINING ADD OF SOURCE LINE.
0F47  F5      PTVAL:  PUSH   PSW
0F48  D5              PUSH   D
0F49  E5              PUSH   H
0F4A  3E 02           MVI    A,002
0F4C  5E              MOV    E,M
0F4D  2C              INR    L
0F4E  56              MOV    D,M
0F4F  2C              INR    L
0F50  D5              PUSH   D
0F51  E3      N1:     XTHL
0F52  5E              MOV    E,M
0F53  23              INX    H
0F54  56              MOV    D,M
0F55  23              INX    H
0F56  E3              XTHL
0F57  73              MOV    M,E
0F58  2C              INR    L
0F59  72              MOV    M,D
0F5A  2C              INR    L
0F5B  3D              DCR    A
0F5C  C2 51 0F        JNZ    N1
0F5F  E3              XTHL
0F60  56              MOV    D,M
0F61  E1              POP    H
0F62  72              MOV    M,D
0F63  E1              POP    H
0F64  D1              POP    D
0F65  F1              POP    PSW
0F66  C9              RET
              ; ROUTINE TO CHK FLAGS ON INPUT AND OUTPUT.
              ; PASSED FLAG VALUE IN REG B.
0F67  F5      MCHK:   PUSH   PSW
0F68  DB 03   MCHK1:  IN     3
0F6A  A0              ANA    B
0F6B  CA 68 0F        JZ     MCHK1
0F6E  F1              POP    PSW
0F6F  C9              RET
              ; MULTIPLICATION ROUTINE (ADD. VALUES)
0F70  5E      MULT:   MOV    E,M
0F71  2B              DCX    H
0F72  56              MOV    D,M
0F73  36 11           MVI    M,11H
0F75  06 00           MVI    B,0
0F77  48              MOV    C,B
0F78  7B      TOP:    MOV    A,E
0F79  1F              RAR
0F7A  5F              MOV    E,A
0F7B  7A              MOV    A,D
0F7C  1F              RAR
0F7D  35              DCR    M
0F7E  57              MOV    D,A
0F7F  C8              RZ
0F80  D2 8D 0F        JNC    SHIFT
0F83  2B              DCX    H
0F84  2B              DCX    H
0F85  78              MOV    A,B
0F86  86              ADD    M
0F87  47              MOV    B,A
0F88  23              INX    H
0F89  79              MOV    A,C
0F8A  8E              ADC    M
0F8B  4F              MOV    C,A
0F8C  23              INX    H
0F8D  79      SHIFT:  MOV    A,C
0F8E  1F              RAR
0F8F  4F              MOV    C,A
0F90  78              MOV    A,B
0F91  1F              RAR
0F92  47              MOV    B,A
0F93  C3 78 0F        JMP    TOP
              ;LINKAGES TO FLOATING POINT ROUTINES
0FCD                  ORG    7715Q
0FCD  C3 80 09        JMP    WZER
0FD0  C3 5C 09        JMP    LADD
0FD3  C3 5F 0A        JMP    LMUL
0FD6  C3 00 09        JMP    LDIV
0FD9  C3 60 09        JMP    LSUB
0FDC  C3 CE 0A        JMP    DFXL
0FDF  C3 57 0A        JMP    LMCM
0FE2  C3 02 0F        JMP    COPY
0FE5  C3 9A 0C        JMP    CVRT
0FE8  C3 21 0F        JMP    INPUT
0FEB  C3 70 0F        JMP    MULT
0FEE  C3 47 0F        JMP    PTVAL
0FF1  C3 3A 0F        JMP    DCOMP
0FF4  C3 67 0F        JMP    MCHK
0FF7  C3 40 0F        JMP    CHAR2
                      END
NO PROGRAM ERRORS
```

Branch to page 120

DIODES/ZENERS

1N914	100v	10mA	.05
1N4004	400v	1A	.08
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.03
1N753A	6.2v	z	.25
1N758A	10v	z	.25
1N759A	12v	z	.25
1N4733	5.1v	z	.25
1N5243	13v	z	.25
1N5244B	14v	z	.25
1N5245B	15v	z	.25

SOCKETS/BRIDGES

8-pin	pcb	.25	ww	.45
14-pin	pcb	.25	ww	.40
16-pin	pcb	.25	ww	.40
18-pin	pcb	.25	ww	.75
22-pin	pcb	.45	ww	.75
24-pin	pcb	.35	ww	1.25
28-pin	pcb	.35	ww	1.45
40-pin	pcb	.50	ww	1.95

Molex pins .01 To-3 Sockets .25

2 Amp Bridge 100-prv 1.20

25 Amp Bridge 200-prv 2.50

TRANSISTORS, LEDS, etc.

2N2222	NPN			.10
2N2907	PNP			.15
2N3740	PNP	1A	60v	.25
2N3906	PNP			.10
2N3055	NPN	15A	60v	.50

LED Green, Red, Clear .15

D.L. 747 7 seg 5/8" high 1.95

XAN72 7 seg com-anode 1.50

FND 359 Red 7 seg com-cathode 1.00

C MOS

4000	.20
4001	.20
4002	.25
4004	4.95
4006	1.20
4007	.40
4008	1.20
4009	.25
4010	.45
4011	.20
4012	.25
4013	.40
4014	1.10
4015	.95
4016	.35
4017	1.10
4018	1.10
4019	.70
4020	.85
4021	1.35
4022	1.15
4023	.25
4024	.95
4025	.35
4026	1.95
4027	.50
4028	.95
4030	.45
4033	1.95
4034	2.45
4035	1.25
4040	1.35
4042	.95
4043	1.25
4044	.95
4046	1.50
4049	.80
4050	.70
4066	1.35
4069	.40
4071	.35
4082	.45

— T T L —

7400	.15
7401	.15
7402	.20
7403	.25
7404	.15
7405	.25
7406	.45
7407	.55
7408	.25
7409	.15
7410	.15
7411	.25
7412	.30
7413	.65
7414	1.10
7416	.25
7417	.50
7420	.15
7426	.40
7427	.45
7430	.15
7432	.45
7437	.45
7438	.35
7440	.25
7441	1.15
7442	.65
7443	.95
7444	.95
7445	.95
7446	.95
7447	.95
7448	1.20
7450	.25
7451	.25
7453	.25
7454	.25
7460	.40
7470	.45
7472	.45
7473	.35
7474	.40
7475	.45
7476	.20
7480	.65
7483	1.00
7485	1.05
7486	.40
7489	2.50
7490	.40
7491	1.15
7492	.95
7493	.45
7494	1.25
7495	.85
7496	.95
74100	1.85
74107	.45
74121	.40
74122	.55
74123	.55
74125	.45
74132	1.35
74141	1.30
74150	1.00
74151	.95
74153	.95
74154	.75
74156	1.15
74157	.75
74161	1.25
74163	1.25
74164	.95
74165	1.50
74166	1.35
74175	.95
74176	1.25
74180	.85
74181	3.25
74182	.95
74190	1.75
74192	1.65
74193	.85
74194	1.45
74195	.95
74196	1.50
74197	1.25
74198	2.35
74367	.85
75108A	.35
75110	.35
75491	.50
75492	.50
74H00	.25
74H01	.25
74H04	.25
74H05	.25
74H15	.30
74H20	.30
74H22	.40
74H30	.25
74H40	.25
74H52	.15
74H53J	.25
74H55	.25
74H72	.55
74H101	.75
74H103	.75
74H106	.95
74L00	.35
74L02	.35
74L03	.30
74L10	.35
74L30	.45
74L47	1.95
74L75	.55
74S00	.55
74S02	.55
74S03	.50
74S10	.45
74S11	.45
74S20	.50
74S40	.30
74S51	.45
74S64	.30
74S74	.50
74S112	1.50
74S133	.45
74S140	.75
74S151A	.45
74S153	.45
74S158	.45
74S194	1.50
74S257 (8123)	.25
74LS00	.45
74LS01	.45
74LS02	.45
74LS04	.55
74LS08	.45
74LS09	.45
74LS10	.45
74LS11	.45
74LS20	.50
74LS21	.25
74LS22	.25
74LS32	.55
74LS37	.40
74LS40	.55
74LS42	1.75
74LS52	1.45
74LS74	.95
74LS90	1.30
74LS93	1.00
74LS107	.95
74LS153	1.20
74LS157	.85
74LS164	1.90
74LS368	.70

9000 SERIES

9301	1.00
9309	.45
9602	1.50

MEMORY, CLOCKS

74S188 (8223)	3.00
8080	26.50
MM1702A	10.50
MM5314	3.50
MM5316	3.95
2102-1	1.75
2102-L1	1.95
TR 1602A	6.95

LINEARS, REGULATORS, etc.

MCT2	.95
8038	3.95
LM201AH	.75
LM301AH	.25
LM308AH	1.00
LM309H	.65
LM309K	.90
LM310	1.15
LM311	1.35
LM320K5	1.65
LM320K12	1.65
LM320T12	1.65
LM320T15	1.65
LM339	1.65
7805 (340T-5)	1.00
LM340T-12	1.25
LM340T-15	1.25
LM340T-18	1.65
LM340T-24	1.25
LM340K-12	2.15
LM340K-15	1.65
LM340K-18	1.65
LM340K-24	1.25
LM373	1.95
LM380	.95
LM709	.30
LM711	.45
LM723	.45
LM725	1.95
LM739	1.50
LM741 8-14	.25
LM747	1.10
LM1307	1.25
LM1458	.95
LM3900	.65
LM75451	.65
NE555	.50
NE556	1.10
NE565	.95
NE566	1.75
NE567	1.35
SN72720	.35
SN72820	.35

INTEGRATED CIRCUITS UNLIMITED

7889 Clairemont Mesa Blvd. • San Diego, CA 92111 • (714) 278-4394

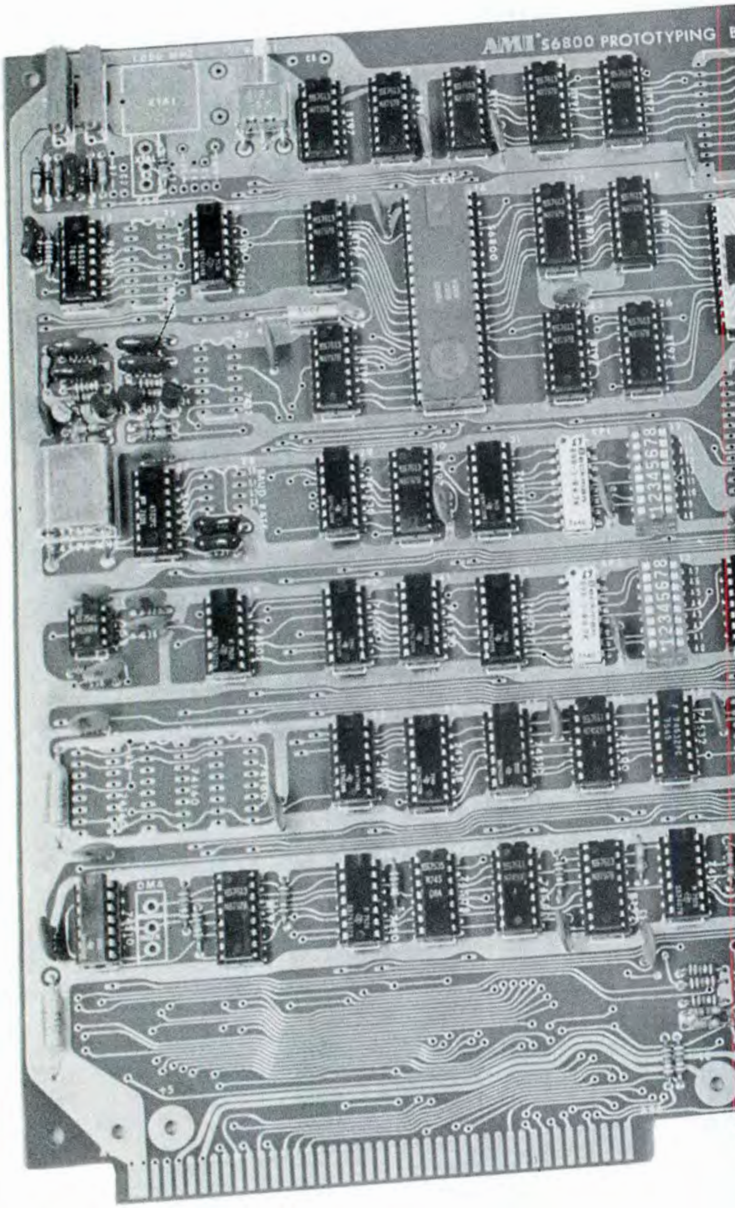
All orders shipped prepaid   No minimum
Open accounts invited   COD orders accepted

Discounts available at OEM Quantities
California Residents add 6% Sales Tax

24 Hour Phone (714) 278-4394   **MasterCharge / BankAmericard**

AMI's EVK SERIES MICROCOMPUTER PROTOTYPING BOARDS

By Robert A. Stevens



INTRODUCTION

This article is Part Three of a series on the EVK Microcomputer hardware, firmware and supporting software. This month's subject covers the ROM resident Prototyping TTY MONITOR Operating System, PROTO.

PROTO SOFTWARE

The resident PROTO software program includes the following commands:

- **L** **LOAD HEX** paper tape program into RAM memory
- **P** **PUNCH HEX** paper tape from memory
- **S** **SET** (write) specified data string characters into consecutive memory locations
- **D** **DISPLAY** (prints) in HEX to TTY contents of specified memory locations
- **G** **GO TO** user program at specified address and execute
- **R** **PRINTS** contents of MPU register (C, B, A, X, P & S) on TTY at time the user's program was last interrupted
- **B** **BURN** (copies) the contents of specified memory into the EPROM in the programming socket
- **V** **VERIFY** (compares) contents of specified memory with EPROM or ROM in the programming socket
- **I** **INPUT** (copies) contents of the EPROM or ROM in the programming socket into specified RAM memory locations
- **M** **MOVE** (copies) contents of memory block from specified location to designated RAM memory location
- **E** **END** of transmission (EOT) character terminates the end of punch paper tape record and punches EOT on paper tape.

The commands will operate on a single character OP CODE plus address parameters from the TTY keyboard.

PROTO COMMAND DESCRIPTIONS

The EVK 300 board will be supplied with a prototyping operating system program (PROTO). The program resides in ROM with a starting address of F000. The various routines within PROTO are called by entering via the TTY keyboard one of the commands described in the following paragraphs. A command consists of one character command identifier followed by additional parameters, if needed, separated by blanks or commas. All commands end with a carriage

Good Software and Support are to a computer as the driver is to his car. One without the other and you have a magnificent paperweight.

return. Since no action is taken before the carriage return, an input line may be deleted by the use of the TTY ESCAPE key.

L, ADDL, ADDH, OFFSET

The Load tape command loads data from a hex formatted tape (see paragraph on 6800 HEX tape format at end of article) into the user's memory between ADDL and ADDH, inclusive. The OFFSET is added to the memory address specified on the tape to form the actual memory starting address for the data stored. If a byte to be stored into memory has an address outside of the range ADDL, ADDH, it is not entered into memory, but a Delete character (H'FF) is transmitted to the terminal.

Example: L 0100 02FF FFFA

The address range in the L command is optional, and if omitted is assumed to be the full range of memory (0000-FFFF). The offset parameter is also optional, and if omitted is assumed to be zero (0000). Thus the L command with no parameters loads the tape into the memory locations specified on the tape with no offset. The offset value in the L command is a two's complement signed number, entered in unsigned hexadecimal. For example, an offset of −6 is entered as FFFA.

If an attempt is made to load non-existent memory, or ROM, the loading operation will terminate, typing out the address and the message "BAD ADR."

In operating the Load command, PROTO turns on the tape reader and scans the tape for the first ASCII "S," which indicates start of record. It is not necessary to position the tape at the first record of a tape file since each record contains its own starting address.

PROTO will load data records until it encounters an end of file (EOF) record or a tape error (Check Sum or illegal character). When PROTO reads a header record (start of record and address), it translates the header into ASCII characters and prints the result. The Check Sum is the binary sum of all characters in the block.

PROTO does not list the tape contents as the tape is being read.

When PROTO encounters an end of file record or a tape error, it turns off the reader and prints "EOF" or "CKSM ERR" respectively.

P, ADDL, ADDH, OFFSET

The Punch hex format command causes PROTO to punch on the TTY paper tape the contents of memory between ADDL and ADDH, inclusive. Each record is punched with a four-digit hex address of the starting byte of the record. This address is derived from the memory address of the byte being punched, plus the offset value, OFFSET. The offset is optional, and if omitted is assumed to be zero.

All data records are punched in hex format. Records using this command (except the last record) contain 16 bytes of data plus the start code, byte count, address, and the checksum.

The P command does not cause an EOF record to be punched so that several disjoint blocks of memory can be combined on one tape file.

Example: P F000 F07F 0F00

S, ADDR, BYTE1, BYTE2, ———, BYTEN

The Set memory command writes the 8-bit data words specified by BYTE1 to BYTEN into consecutive memory locations starting at ADD.

If ADD has more than 4 (hexadecimal) characters or if any of the data bytes have more than 2 characters each, only the last 4 or 2 characters are used respectively.

Example: S 0000 86 05 97 28

Memory locations at 0000 through 0003 are loaded as shown.

D, ADDL, ADDH

The Display memory command prints the contents of memory between ADDL and ADDH, inclusive, in hex format. Up to sixteen bytes per line are printed, preceded by the hexadecimal address of the first byte of the line. A carriage return is forced after a byte having a low order digit of F in its memory address is printed.

Example: D FC00 FC1F

Two lines of memory contents are printed as follows:

```
FC00 00 01 02 03 04 . . . 0E0F
FC10 10 11 12 13 14 . . . 1E1F
```

G, ADDR

The Go command starts execution of the user program at the address specified by the input parameter. To insure that all registers contain the same information they held before the user program was interrupted, PROTO pushes into the stack the copy of the user registers that it keeps at locations FFEB—FFF3 (CC, B, A, X, P, S) then executes an RTI instruction. The user can change the initial values of the

registers by changing the contents of these locations.

Example: G 300

Program will branch to address 0300 and start execution from that point.

R

The Registers command prints the contents of memory locations FFEF—FFF3 which contain the values that were in the user's C, B, A, X, P, and S registers (in that order) when the user's program was last interrupted.

B, ADDL, ADDH, ROMAD

The Burn command copies the contents of user memory into the EPROM in the programming socket, beginning with memory location ADDL through ADDH, inclusive, to EPROM locations beginning with address ROMAD. Each byte is burned in with 20 3-ms pulses of −50V on the V_{PROG} pin (pin 11) of the EPROM. Before attempting to write into the EPROM, the contents of the EPROM are compared with the user memory data byte to verify that the EPROM will take the byte (PROTO will not attempt to program a EPROM location to logic LOW which already contains logic HIGH). After the 20 pulses, the new contents of the EPROM are verified against the memory byte to be sure the data was indeed written. If the byte did not program, a NAK code is typed out on the terminal, and another try is made, up to a maximum of three tries.

If the preverify encounters a EPROM location containing HIGHs where the memory byte has zeros, PROTO will type out the memory address, the memory byte in binary, the EPROM byte in binary, and the EPROM address (if different from the memory address), then stop. If after attempting to write data into the EPROM, the data does not program, or erroneous bits show up, a similar display occurs for the failing location, with the additional message "BAD ADR" typed on the same line.

The EPROM address ROMAD is optional, and if omitted, ADDL is used, with only the least significant nine bits of the address being used. If the address range ADDL, ADDH is omitted, the 512 bytes beginning at FC00 are used, and the EPROM is checked to insure it contains all LOWs before any locations are written. If not, four question marks are typed and the B command is aborted.

V, ADDL, ADDH, ROMAD

The Verify command compares user memory between ADDL and ADDH, inclusive, with the corresponding locations in the EPROM in the prgramming socket, beginning with EPROM address ROMAD. Each location that does not match is typed out in the following format:

aaaa mmmmmmmm pppppppp rrrr

where "aaaa" represents the user memory address, "mmmmmmmm" represents the memory byte, in binary, and "rrrr" represents the EPROM address, if different from the memory address (in the low nine bits). Nothing is typed for matching locations. The typeout may be aborted by typing an ESC key during the typeout.

If the ROMAD parameter is omitted, ADDL is assumed. If no parameters are supplied in the command, the whole EPROM is compared to the contents of FC00 — FDFF.

I, ADDL, ADDH, ROMAD

The Input command copies the contents of an EPROM in the programming socket into memory beginning at the address ADDL through ADDH, inclusive, from the EPROM address ROMAD. If ROMAD is omitted, ADDL is assumed. If no parameters are supplied, the entire EPROM is copied into the RAM area, FC00 — FDFF. An attempt to copy an EPROM into non-existent memory will abort the command with the message "BAD ADR."

M, ADDL, ADDH, DEST

The Move command copies memory from the range ADDL — ADDH, inclusive, to the RAM locations starting at DEST. This copy begins at the lower address, so if DEST lies within the range ADDL — ADDH, some of the original data will be lost, and other parts will be duplicated.

E

The End of Transmission command is used to cause an EOT character to be punched on the paper tape. After a field has been punched, an EOT will terminate the record and punch a trailer tape. When reading a record, the reader will stop at the EOT character. If no EOT character is present, the reader must be manually turned off and the Reset switch must be pressed to enter the operating system program.

THE SUBROUTINE ROM

Many of the monitor's functions are accomplished with the help of the Re-Entrant Self-Relative Subroutine ROMs $(RS)^3$. This standard ROM, which can be considered a software extension to the 6800 instruction set, is also available to be used by the user both on the prototype board and in his final production system. The user can call one of the 25 $(RS)^3$ subroutines with an SWI instruction followed by the number of the desired subroutine.

The user should be aware of the fact that the $(RS)^3$ pushes from 7 to 10 bytes of data onto the stack, depending upon which subroutines are called. This means that if the user calls $(RS)^3$ routines, he must make sure that the necessary memory space is available for stack expansion.

Since PROTO assigns its own stack area, the user need not be concerned about how $(RS)^3$ is used.

INTERRUPTS

Of the four available interrupt vectors, IRQ, RESET and SWI are used by PROTO while NMI is left for the user. The vectors are in RAM (except for RESET which is switch controlled) so the user writing his own program can completely control the system.

The upper memory locations are RAM. If the user

expects either NMI or IRQ interrupts to occur, he must initialize the vector addresses to the starting address of the IRQ and NMI handler routines.

PROTO must have control of the RESET vector so that the RESET switch on the Prototyping Board can return program control to PROTO at any time.

The reset routine copies the contents of the B, A, X, CC, and S registers into a fixed area of memory. This means that the program can be aborted at any time by using the reset switch while still saving all the registers except the program counter. Unfortunately, the contents of the program counter are lost.

It is possible for the user to use the NMI interrupt to abort a program execution without losing the contents of the P and C registers. This condition is automatically set in the NMI handling routine when PROTO is called. This interrupt vector will cause the contents of the user's registers to be printed when the $\overline{NMI}$ line goes low.

Since the SWI instruction is used to call subroutines between 00 and H'18 from $(RS)^3$ the user is somewhat limited in the ways he can use SWI instructions. However, he can access an SWI handler routine in his own program by an SWI instruction followed by a byte containing the decimal number less than H'80 but greater than H'19 < n < H'80 sequence, PROTO passes control at address FFF4. If the user expects to access his own SWI routine and use PROTO, he must use the Set Memory command to store the address of this routine at locations FFF4 and FFF5.

PROTO makes sure that the user's SWI routine is entered from the stack with all registers containing the same information that they would hold if the routine were entered directly through the SWI vector.

BREAKPOINTS

Breakpoints allow the user to halt his program and examine the contents of the internal registers. PROTO provides two types of breakpoints. In this system, breakpoints are actually debugging routines that can be called from the user's program just like $(RS)^3$ routines.

Each breakpoint requires a two byte calling sequence: and SWI instruction followed by a number.

Breakpoints may be inserted either by reassembling the program with the extra SWI instructions added or the Set Memory command may be used to replace parts of the code with SWI instructions. Note that the second method is not satisfactory for the snapshot option (described below) since the replaced code must be restored before execution can be continued. When using the second method, the user must make sure that he replaces the first two bytes of an instruction. If the SWI replaces the second or third byte of an instruction, it may be interpreted as an address rather than an opcode.

The different types of breakpoints are:

1. Print registers (SWI, H'80)
2. Snapshot (SWI, H'81)

The sequence SWI, H'80 saves the user's registers at the vector stored in FFF4 — FFF5, prints their contents (in the order CC BB AA XXXX PPPP SSSS), then returns control to PROTO.

The sequence SWI, H'81 prints out the contents of the user's registers then continues executing the user's program starting at the address following the byte containing the number H'81. Note that if this address does not contain a valid opcode, unpredictable results will occur.

6800 PAPER TAPE HEX FORMAT

The AMI 6800 Hex Tape format provides a compact representation of binary data patterns for transmission using ASCII communication terminals.

The Hex tape is organized into data records with each record containing information in the same format. The record information consists of type, length, address, data and checksum. All records begin with an 'S' character for start of record identification. All information on the tape which is not between a start of record and the checksum is ignored.

TAPE FORMAT

ASCII Character	Description
1	Start of record (S)
2	Type of record 0 — Header record 1 — Data record 9 — End of file record
3—4	Byte Count Since each data byte is represented as two hex characters, the byte count must be multiplied by two to get the number

	of characters to the end of the record. (This includes checksum and address data.)
5, 6, 7, 8	Address Value The memory location where this record is to be stored.
9,...,N	Data Each data byte is represented by two hex characters.
N+1, N+2	Checksum The one's complement of the additive summation (without carry) of the data bytes, the address, and the byte count.

Example Data Record

Memory Contents

Address	**Data**
A000	10
A001	1A
A002	20
A003	2A

Data Record Contents

Character		**Tape**		
1	Start of record	53	S	
2	Type of record	31	1	
3	Byte count	30	0	Checksum
4		37	7	Checksum
5		41	A	Checksum, Byte Count *2
6		30	0	Checksum, Byte Count *2
7	Address	30	0	Checksum, Byte Count *2
8		30	0	Checksum, Byte Count *2
9	Data byte 1	31	1	Checksum, Byte Count *2
10		30	0	Checksum, Byte Count *2
11	Data byte 2	31	1	Checksum, Byte Count *2
12		41	A	Checksum, Byte Count *2
13	Data byte 3	32	2	Checksum, Byte Count *2
14		30	0	Checksum, Byte Count *2
15	Data byte 4	32	2	Checksum, Byte Count *2
16		41	A	Checksum, Byte Count *2
17	Checksum	38	8	Byte Count *2
18		34	4	Byte Count *2

The format for all hex tape records is diagrammed below.

Character		**Header Record**		**Data Record**		**End-of-File Record**	
1	Start of Record	53	S	53	S	53	S
2	Type of Record	30	0	31	1	39	9
3	Byte Count	31	12	31	16	30	03
4		32		36		33	
5		30		31		30	
6	Address	30	0000	31	1100	30	0000
7	(if any)	30		30		30	
8		30		30		30	
9	Data	34		39	98	46	FC
10		38		38		43	(Checksum)
•		34		30	02		
•		34		32			
•		35					
•		32					
•				41			
•				48	A8 (Checksum)		
N	Checksum	39	9E				
		45					

SEE MICROCOMPUTER SOFTWARE DEPOSITORY PROGRAM INDEX FOR COPIES OF THIS PROGRAM.

PROTO PROGRAM

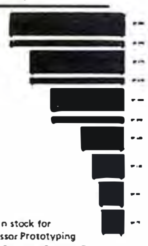
```
PAGE  1  PROTO   01/09/76  9:22  PROTO

LOC   OBJECT  M  SOURCE STATEMENT

                         TITLE  PROTO
                         OPT    LSKP
                 ************************************************
                 *
                 * PROTOTYPE BOARD MONITOR PROGRAM
                 *
                 * VERSION 2.0   01/08/76
                 *
                 * COPYRIGHT 1976 BY AMERICAN MICROSYSTEMS INC.
                 *
                 *
                 ************************************************
                 *
                 * DEFINITIONS
                 *
        FBCE A   ACIAC   EQU    $FBCE        ACIA CONTROL REG
        FBCF A   ACIAD   EQU    $FBCF        ACIA DATA REG
        FBCE A   ACIAS   EQU    $FBCE        ACIA STATUS REG
        0020 A   BLANK   EQU    $20          BLANK CHAR
        000D A   CR      EQU    $0D          CARRIAGE RET CHAR
        0018 A   ESC     EQU    $18          ABORT CHAR
        0004 A   EOT     EQU    $04          END OF MSG TO BE PRINTED
        FFFF A   LAST    EQU    $FFFF        HIGHEST ROM ADDRESS
        000A A   LF      EQU    $0A          LINE FEED
        007F A   RUBOUT  EQU    $7F
                 *
                 *   EXTERNALS FOR RSRSR AND PROM BURNER
                 *
                         DEF    MOVENT,GETRNG,NXTADR,PXISTS,RNGERR,PBADR
                         DEF    PCRLF,OUTCH,PSPACE,SETMEM,ABORT
                         DEF    PROMAD,ADR,ADDL,ADDH,COUNT,MONITR
                         REF    RSRSR,BURN,MOVE,READ,VFY,PINIT
                 *
                 *RSRSR ROUTINE DEFINITIONS:
                 *
        000B A   SUBXAB  EQU    11           SUBTRACT X FROM A,B
        0008 A   ADDABX  EQU    8            ADD A,B TO X
        0012 A   PMSG    EQU    18           PRINT MSG
        000F A   P2HEX   EQU    15           PRINT BYTE AS 2 HEX CHARS
        0010 A   P4HEX   EQU    16           PRINT WORD AS 4 HEX CHARS
        0015 A   CONHB   EQU    21           CONVERT HEX TO BINARY
        0011 A   PUTA    EQU    17           OUTPUT TO ACIA
        0014 A   GETA    EQU    20           INPUT FROM TTY
        0013 A   ALPNUM  EQU    19           TEST FOR ALPHANUMERIC
        0009 A   PRTXD   EQU    9            CONV. X TO DEC. & PRINT
                 *
                 *SUBR IS A MACRO TO CALL RSRSR ROUTINES
                 *
                 SUBR      MACRO  PARAM
                         SWI
                         BYTE  PARAM
                           MEND
                 *
                 ************************************************
                 *
                 * MONITOR RAM
                 *
FF90                     ORG    $FFFE-110    ***CHANGE IF RAM USAGE CHANGES
        FF90 A   BASE    EQU    *            BASE ADR USED WITH INDEX OPS
        FF8F A   BOS     EQU    *-1          BOTTOM OF MONITOR STACK
                 *
FF90    0048     BUF     RMB    72           LINE OF TTY INPUT
                 *
        FFD8 A   PROMAD  EQU    *            ADDRESS IN PROM
FFD8    0002     OFFSET  RMB    2            OFFSET FOR LOADER/PUNCH
FFDA    0002     ADR     RMB    2            PARAM. ENTERED BY USER
FFDC    0002     ADDL    RMB    2
FFDE    0002     ADDH    RMB    2
FFE0    0002     BUFPTR  RMB    2            POINTER TO LAST CHAR SCANNED
FFE2    0001     RECTYP  RMB    1            TAPE RECORD TYPE
FFE3    0001     COUNT   RMB    1            COUNT FIELD FROM TAPE
FFE4    0001     CKSM    RMB    1            CALCULATED CKSM
FFE5    0002     SAVESP  RMB    2            TEMP STORAGE FOR S REG
FFE7    0002     SAVEX   RMB    2            TEMP STORAGE FOR X REG
FFE9    0001     ECHO    RMB    1            1=ECHO TTY, 0=NO ECHO
FFEA    0001     TCOUNT  RMB    1            TEMP LOC FOR COUNT
                 * USER REGISTERS
FFEB    0001     CREG    RMB    1
FFEC    0001     BREG    RMB    1
FFED    0001     AREG    RMB    1
FFEE    0002     XREG    RMB    2
FFF0    0002     PREG    RMB    2
FFF2    0002     SPEG    RMB    2
                 *
FFF4    0002     USWI    RMB    2            USER SWI VECTOR (MAY NOT BE IMPLEMENTED)
FFF6    0002     ACIAI   RMB    2            INDIRECT POINTER TO ACIA FOR RSRSR
FFF8    0002     IRQVEC  RMB    2            INTERRUPT REQUEST VECTOR
FFFA    0002     SWIVEC  RMB    2            SOFTWARE INTERRUPT VECTOR
FFFC    0002     NMIVEC  RMB    2            NON-MASKABLE INTERRUPT VECTOR
                 ************************************************
                 *
                 *   *** MONITOR ENTRY VECTOR ***
                 *
                 * RESTART INTERRUPT HANDLER
                 * INTERRUPT BREAK HANDLER
                 *
                 ************************************************
0000                     ISEC
        0000 I   START   EQU    *            RESET INTERRUPT HANDLER
0000  20 05              BRA    START1
        0002 I   BREAK   EQU    *            BREAK ON INTERRUPT ROUTINE
0002  7E 0087 I          JMP    BREAK1
0005    FBCE A   ACIAA   WORD   ACIAC        POINTER TO ACIA
                 *
        0007 I   START1  EQU    *
0007  36                 PSH A               SAVE A REG IF STACK EXISTS
0008  07                 TPA                 SAVE CONDITION CODES
0009  B7 FFEB A          STA A  CREG
000C  32                 PUL A
000D  B7 FFED A          STA A  AREG         SAVE CURRENT VALUE OF REGS
0010  F7 FFEC A          STA B  BREG
0013  FF FFEE A          STX    XREG         SAVE X
0016  BF FFF2 A          STS    SPEG         SAVE SP
0019  8E FF8F A          LDS    #BOS         INIT. SREG TO MON. STACK
001C  CE 0002 I          LDX    #BREAK       BREAKPOINT ROUTINE
001F  FF FFFC A          STX    NMIVEC       STORE IN INTERRUPT VECTORS
0022  FF FFF8 A          STX    IRQVEC
0025  CE 00D1 I          LDX    #SWI30
0028  FF FFF4 A          STX    USWI
002B  CE 00BE I          LDX    #SWIHAN      SOFTWARE INTERRUPT HANDLER
002E  FF FFFA A          STX    SWIVEC
0031  CE 0005 I          LDX    #ACIAA       SET UP ACIA PTR
0034  FF FFF6 A          STX    ACIAI
0037  86 03              LDA A  #3           RESET ACIA
0039  B7 FBCE A          STA A  ACIAS
                 *
003C  86 01              LDA A  #1           SET ACIA CR
003E  B7 FBCE A          STA A  ACIAC
                 * PRINT CR,LF, & RETURN TO MONITOR
                 *
        0041 I   MONENT  EQU    *
        0041 I   MONEN1  EQU    *
0041  BD 0005 R          JSR    PINIT
0044  BD 0304 I          JSR    PCRLF
                 *
                 ************************************************
                 *
                 *
                 * MONITOR ENTRY POINT
                 *
                 ************************************************
        0047 I   MONITR  EQU    *
0047  8E FF8F A          LDS    #BOS         INIT MON. STACK
004A  BD 0398 I          JSR    RDROFF       TURN OFF READER
004D  B6 FBCF A          LDA A  ACIAD        DUMP TTY INPUT DATA
0050  86 3E              LDA A  #'>          PROMPT USER
0052  BD 0200 I          JSR    OUTCH
                 *
                 * READ TTY LINE (BUFPTR)
                 *  STORE TTY INPUT IN BUF UNTIL CR IS HIT
                 *
0055  CE FF90 A          LDX    #BUF         INITIALIZE BUFPTR
0058  FF FFE0 A          STX    BUFPTR
005B  0D                 SEC                 SET ECHO FLAG
005C  79 FFE9 A          ROL    ECHO
                 *BEGIN UNTIL LOOP
005F  8C FFD7 A  RT10    CPX    #BUF+71      TEST FOR BUF OVERFLOW
0062  26 02              BNE    RT20         NO OVERFLOW
0064  20 47              BRA    ABORT
0066  BD 0400 I  RT20    JSR    WAITTY       READ NEXT CHAR
0069  A7 00      RT30    STA A  0,X          INSERT CHAR INTO BUF
006B  08                 INX                 INC BUFPTR
                 * WHILE CONDITION :
006C  81 0D      RT90    CMP A  #CR          CARRIAGE RETURN ?
006E  26 EF              BNE    RT10         NO, CONTINUE LOOP
                 *END OF LOOP
                 *
                 * DECODE 1 CHAR COMMAND
                 *   COMPARE CHAR WITH TABLE OF VALID CHARS FOLLOWED BY
                 *   ADDRESSES OF APPROPRIATE ROUTINES.
                 *
0070  BD 0385 I          JSR    PXISTS       GET 1ST CHAR
0073  08                 INX                 INC BUFPTR
0074  FF FFE0 A          STX    BUFPTR
0077  CE 008C I          LDX    #CTABLE      START OF TABLE
                 * BEGIN LOOP
007A  A1 00      DLOOP   CMP A  0,X          COMPARE
007C  26 04              BNE    DL10
                 * FOUND CHAR.  GET ADDRESS IMMEDIATELY FOLLOWING CHAR.
007E  EE 01              LDX    1,X
0080  6E 00              JMP    0,X          GO TO PROPER ROUTINE
                 * NO COMPARE.  MOVE TO NEXT CHAR.
0082  08         DL10    INX
0083  08                 INX
0084  08                 INX
0085  8C 00AD I          CPX    #CTEND       END OF TABLE?
0088  26 F0              BNE    DLOOP        NO. REPEAT
                 * END LOOP.
008A  20 21              BRA    ABORT        NOT IN TABLE.
                 *
        0047 I   MONEND  EQU    MONITR
                 *
                 *
                 * CTABLE:  TABLE OF VALID 1 CHARACTER COMMANDS.
                 *      EACH ENTRY CONSISTS OF 3 BYTES.  BYTE 1
                 *    CONTAINS THE ASCII CHAR.  BYTES 2,3 CONTAIN THE
                 *    ADDRESS OF THE APPROPRIATE ROUTINE.
                 *
        008C I   CTABLE  EQU    *
008C  4C                 BYTE   'L
008D  01AD I             WORD   LOAD
008F  47                 BYTE   'G
0090  019F I             WORD   GO
0092  50                 BYTE   'P
0093  0308 I             WORD   PUNCH
0095  42                 BYTE   'B
0096  0001 R             WORD   BURN
0098  4D                 BYTE   'M
0099  0002 R             WORD   MOVE
009B  56                 BYTE   'V
009C  0004 R             WORD   VFY
009E  49                 BYTE   'I
009F  0003 R             WORD   READ
00A1  53                 BYTE   'S
00A2  03E3 I             WORD   SM
00A4  44                 BYTE   'D
00A5  0136 I             WORD   DM
00A7  52                 BYTE   'R
00A8  00EE I             WORD   PREGS
00AA  45                 BYTE   'E
00AB  0150 I             WORD   EOF
        00AD I   CTEND   EQU    *
                 *
                 ************************************************
                 *
                 * ABORT
                 *
                 ************************************************
                 *
        00AD I   ABORT   EQU    *
        00AD I   BAUINP  EQU    *
00AD  CE 0273 I          LDX    #MQUES       PRINT ????
                 *
                 * PRINT MSG AND RETURN TO MONITOR
                 *
        00B0 I   MSGMON  EQU    *
        00B0 I   MSGABT  EQU    *
00B0  8E FF8F A          LDS    #BOS         S:=BOTTOM OF STACK
00B3                     SUBR   PMSG
00B5  20 8A              BRA    MONEN1
                 *
                 ************************************************
                 *
                 * SWI HANDLER:
                 *    DETERMINE WHETHER SWI IS MONITOR CALL, RSRSR CALL,
                 *    OR USER SWI (NOT IMPLEMENTED).
                 *
                 ************************************************
                 *
        0087 I   BREAK1  EQU    *            BREAKPOINT ENTRY
0087  BD 0005 R          JSR    PINIT        CLEAR PROM BURNER
008A  86 80              LDA A  #128         PRETEND TO BE SWI 128
008C  20 1A              BRA    SWI40        SAVE REGS
                 *
        00BE I   SWIHAN  EQU    *
                 * FIND INDEX BYTE (BYTE AFTER SWI THAT GOT US HERE)
00BE  30                 TSX
00BF  EE 05              LDX    5,X          X:=RET. ADR.
00C1  A6 00              LDA A  0,X          A:=INDEX BYTE
00C3  2B 0C              BMI    SWI30        BREAKPOINT?
                 * IF USER HAS ADDITIONAL (RS)**3 ADDR OF FIRST+2 MUST BE IN FFF4
00C5  80 18              SUB A  #24          RSRSR CALL?
00C7  2A 03              BPL    SWI20        NO --
00C9  7E 0000 R          JMP    RSRSR
                 *
                 *  USER SWI
                 *
00CC  FE FFF4 A  SWI20   LDX    USWI
00CF  6E 00              JMP    0,X          GO DO IT
                 *
                 * MONITOR CALL.  COPY REGS FROM STACK
                 *
00D1  30         SWI30   TSX                 INCREMENT RET. ADDR.
00D2  6C 06              INC    6,X
00D4  26 02              BNE    SWI40
00D6  6C 05              INC    5,X
00D8  CE FFEB A  SWI40   LDX    #CREG        DEST. FOR 1ST REG
                 * BEGIN LOOP
00DB  33         SWI50   PUL B               GET REG
00DC  E7 00              STA B  0,X          COPY
00DE  08                 INX                 MOVE TO NEXT REG
00DF  8C FFF2 A          CPX    #CREG+7      END OF LOOP?
00E2  26 F7              BNE    SWI50
                 * END LOOP
                 *
                 * S NOW CONTAINS ITS VALUE BEFORE SWI
                 * WAS EXECUTED.  SAVE IT.
                 *
```

```
00E4  AF 00              STS    0,X
                  *
                  * A STILL CONTAINS SWI INDEX.  TEST IT
                  *
00E6  81 81              CMP A  #129
00E8  26 04              BNE    PREGS         NOT 129: BREAK
00EA  8D 07              BSR    PR1           129:  SNAPSHOT
00EC  20 1E              BRA    RESTAK        AND RETURN TO USER PROGRAM
                  *
                  ***********************************************
                  *
                  * PREGS:  PRINT USER REGISTERS
                  *
                  ***********************************************
                  *
         00EE I   PREGS  EQU    *
00EE  8D 03              BSR    PR1
00F0  7E 0047 I          JMP    MONEND
         00F3 I   PR1    EQU    *             SUBROUTINE TO PRINT REGS
00F3  CE FFE8 A          LDX    #CREG         X POINTS TO 1ST BYTE OF AREA
                  * PRINT 3 1-BYTE REGS
00F6  C6 03              LDA B  #3            SET UP COUNT
                  *
00F8              PR10   SUBR   P2HEX
00FA  BD 0380 I          JSR    PSPACE
00FD  5A                 DEC B
00FE  2E F8              BGT    PR10
                  *
                  * PRINT 3 2-BYTE REGS
0100  C6 03              LDA B  #3            SET UP COUNT
                  *
0102  BD 037C I   PR20   JSR    P4HEXS
0105  5A                 DEC B
0106  2E FA              BGT    PR20
                  *
0108  BD 0304 I          JSR    PCRLF         PRINT CRLF
010B  39                 RTS                  RETURN
                  *
                  ***********************************************
                  *
                  * RESTORE USER STATUS AND RETURN FROM MONITOR
                  *
                  ***********************************************
                  *
                  * RESTORE USER'S  STATUS
                  *
010C  BE FFE2 A   RESTAK LDS    SREG          TOP OF USER STACK
010F  CE FFE1 A          LDX    #CREG+6       USER REGS.
                  *BEGIN LOOP
                  RUS10
0112  A6 00              LDA A  0,X           GET USER REG
0114  36                 PSH A                PUSH INTO USER STACK
0115  09                 DEX                  MOVE TO NEXT REG
0116  8C FFEA A          CPX    #CREG-1       LAST REG ?
0119  26 F7              BNE    RUS10         NO, CONTINUE LOOP
                  *END OF LOOP
011B  3B                 RTI                  RETURN TO USER PROG
                  ***********************************************
                  *
                  * COMMANDS AND SUBROUTINES:
                  *
                  ***********************************************
                  *
                  ***********************************************
                  *
                  * CHEKSM(CKSM)
                  *   VALIDATE CKSM
                  *
                  ***********************************************
         011C I   CHEKSM EQU    *
011C  B6 FFE4 A          LDA A  CKSM          SAVE CALC. CKSM
011F  36                 PSH A
0120  BD 029E I          JSR    NEXT2D        A:= NEXT BYTE FROM TAPE
0123  33                 PUL B
0124  53                 COM B                B:=CALC. CKSM
0125  11                 CBA                  B=TAPE CKSM?
0126  26 01              BNE    CS1           NO.
0128  39                 RTS
                  *
0129  30          CS1    TSX                  X:=ADR OFCALC. CKSM
012A  09                 DEX
012B                     SUBR   P2HEX         PRINT CALC. CKSM
012D  BD 0380 I          JSR    PSPACE
0130  CE 0278 I          LDX    #MCSER        PRINT "CKSM ERR"
0133  7E 0080 I          JMP    MSGABT
                  *
                  ***********************************************
                  *
                  * DM  ADDL,ADDH   COMMAND
                  *
                  ***********************************************
         0136 I   DM     EQU    *
0136  8D 35              BSR    GETRNG        GET ADD RANGE FROM BUF
                                          RETURNS ADDL,AUDH+1
                  *
                  *BEGIN OUTER LOOP
0138  CE FFDC A   DM10   LDX    #ADDL
013B  BD 037C I          JSR    P4HEXS        PRINT ADDL, SPACE
                  * BEGIN INNER LOOP
013E  FE FFDC A   DM20   LDX    ADDL
0141                     SUBR   P2HEX         PRINT MEM(X),SPACE,INC X
0143  BD 0380 I          JSR    PSPACE
0146  FF FFDC A          STX    ADDL
0149  BC FFDE A          CPX    ADDH          IF ADDL=ADDH+1, END OF RANGE
014C  27 0C              BEQ    DM50          EXIT OUTER LOOP
014E  B6 FFDD A          LDA A  ADDL+1        IF LSB'S OF ADDL=0, END OF LINE
0151  84 0F              AND A  #$F
0153  26 E9              BNE    DM20          NOT END OF LINE. CONTINUE
                  * END OF INNER LOOP
0155  BD 0304 I          JSR    PCRLF         PRINT CR,LF
0158  20 DE              BRA    DM10          EXIT INNER LOOP
                  * END OF OUTER LOOP
015A  7E 0041 I   DM50   JMP    MOVEN1        CR,LF, BACK TO MONITOR
                  *
                  ***********************************************
                  *
                  ** PUNCH END OF FILE AND 60 NULLS
                  *
                  ***********************************************
015D  CE 028A I   EOF    LDX    #MPEOF        PUNCH EOF RECORD
0160                     SUBR   PMSG
                  *
                  ***********************************************
                  *
                  ** PUNCH END OF FILE AND 60 NULLS
                  *
                  ***********************************************
015D  CE 028A I   EOF    LDX    #MPEOF        PUNCH EOF RECORD
0160                     SUBR   PMSG
                  *
                  * PUNCH 60 NULLS
                  *
0162  C6 3B       NULLS  LDA B  #59           LOAD COUNTER
                                          BEGIN LOOP
                  *
0164  4F          NULL1  CLR A                LOAD NULL
0165  BD 020D I          JSR    OUTCH         PRINT ONE NULL
0168  5A                 DEC B                DECREMENT COUNTER
0169  26 F9              BNE    NULL1         DONE?
                  *                       END OF LOOP
016B  20 ED              BRA    DM50          CR/LF,BACK TO MONITOR
                  ***********************************************
                  *
                  * GETRANGE (ADDL,ADDH,BUFPTR)
                  *   GET ADDRESS RANGE FROM BUF
                  *   ABORT IF INVALID
                  *     SET ADDH:=ADDH+1 TO SIMPLIFY COMPARISONS
                  *   RETURNS ADDL & ADDH+1
                  *   ALTERS ADR,X,A,B
                  *
                  ***********************************************
         016D I   GETRNG EQU    *
016D  BD 0288 I          JSR    NXTADR        GET ADDL
0170  FE FFDA A          LDX    ADR
0173  FF FFDC A          STX    ADDL          STORE ADDL
0176  FF FFDE A          STX    ADDH          MAY BE ONLY 1 PARAM
0179  BD 0288 I          JSR    NXTADR        GET ADDH
017C  27 06              BEQ    GETRG3        ONLY 1 PARAM
                  *
017E  FE FFDA A   GETRG1 LDX    ADR
0181  FF FFDE A          STX    ADDH          SAVE ADDH
                  * THE NEXT 5 INSTR TEST ADDH>ADDL
0184  CE FF90 A   GETRG3 LDX    #BASE         REF W.R.T. BASE OF RAM
0187  A6 4E              LDA A  ADDH-BASE,X   MSBYTE
0189  E6 4F              LDA B  ADDH+1-BASE,X
018B  E0 4D              SUB B  ADDL+1-BASE,X
018D  A2 4C              SBC A  ADDL-BASE,X
018F  24 06              BCC    GETRG4        ADDH.GE.ADDL
0191  CE 0265 I   RNGERR LDX    #MRNGER       RANGE ERR MSG
0194  7E 0080 I          JMP    MSGABT        PRINT MSG & ABORT
                  *
0197  FE FFDE A   GETRG4 LDX    ADDH          INC ADDH
019A  08                 INX
019B  FF FFDE A          STX    ADDH
019E  39                 RTS
                  *
                  *
                  ***********************************************
                  *
                  * GO COMMAND
                  *
                  ***********************************************
                  *
019F  BD 0288 I   GO     JSR    NXTADR        GET PARAM
01A2  27 06              BEQ    G10           NO PARAM. CONTINUE EXECUTION
                  *
01A4  FE FFDA A          LDX    ADR           ADR=PARAM FROM NXTADR
01A7  FF FFF0 A          STX    PREG
                  *
01AA  7E 010C I   G10    JMP    RESTAK        (IN INTERRUPT HANDLER)
                  *
                  *
                  ***********************************************
                  *
                  * LOAD COMMAND
                  *
                  ***********************************************
                  *
         01A0 I   LOAD   EQU    *
01AD  CE 0000 A          LDX    #0            INITIALIZE RANGE & OFFSET
01B0  FF FFD8 A          STX    OFFSET        TO 0000-FFFF,0000
01B3  FF FFDC A          STX    ADDL
01B6  09          LODFST DEX
01B7  FF FFDE A          STX    ADDH
01BA  BD 0288 I          JSR    NXTADR        ANY OPERANDS?
01BD  27 1E              BEQ    LHF2          NO, USE DEFAULT.
01BF  FE FFDA A          LDX    ADR           YES.
01C2  FF FFD8 A          STX    OFFSET        IF ONE, IT'S OFFSET
01C5  BD 0288 I          JSR    NXTADR        ANOTHER?
01C8  27 13              BEQ    LHF2          NO.
01CA  FE FFD8 A          LDX    OFFSET        YES.  FIRST TWO ARE RANGE
01CD  FF FFDC A          STX    ADDL
01D0  CE 0000 A          LDX    #0
01D3  FF FFD8 A          STX    OFFSET
01D6  8D A6              BSR    GETRG1
01D8  FE FFDE A          LDX    ADDH
01DB  20 D9              BRA    LODFST        GO TRY AGAIN FOR OFFSET
                  * BEGIN OUTER LOOP
01DD  BD 03A5 I   LHF2   JSR    RDRON         TURN ON READER
                  * SHORT LOOP TO SKIP HDR RECORDS
01E0  8D 70       RDPRE  BSR    FINDS         FIND START OF RECORD
                  *                       SETS (ECHO):=0  ON ENTRY
01E2  BD 0400 I          JSR    WAITTY        RETURNS (A):=TTY I/P
01E5  81 30              CMP A  #'0           IGNORE HDR RECORDS
01E7  27 F7              BEQ    RDPRE
                  * END SHORT LOOP
01E9  B7 FFE2 A          STA A  RECTYP        SAVE RECORD TYPE
01EC  7F FFE4 A          CLR    CKSM
01EF  BD 029E I          JSR    NEXT2D        READ BYTE COUNT FROM TAPE
01F2  4A                 DEC A                DEDUCT ADR & CKSM
01F3  4A                 DEC A
01F4  4A                 DEC A
01F5  B7 FFE3 A          STA A  COUNT         SAVE BYTE COUNT
01F8  BD 029E I          JSR    NEXT2D        READ ADR FIELD FROM TAPE
01FB  B7 FFDA A          STA A  ADR           1ST BYTE
01FE  BD 029E I          JSR    NEXT2D
0201  BB FFD9 A          ADD A  OFFSET+1
0204  B7 FFDB A          STA A  ADR+I         2ND BYTE
0207  B6 FFDA A          LDA A  ADR           CARRY TO FIRST BYTE
020A  B9 FFD8 A          ADC A  OFFSET
020D  B7 FFDA A          STA A  ADR
0210  B6 FFE2 A          LDA A  RECTYP        GET RECORD TYPE (0,1,9)
0213  81 31       LHF3   CMP A  #'I           DATA RECORD ?
0215  26 14              BNE    LHF4          NO
                  *
                  * LOAD DATA RECORD
                  *
                  *BEGIN UNTIL LOOP
0217  BD 029E I   LDR10  JSR    NEXT2D        READ 2 HEX DIGITS FROM
                  *                       TAPE. RETURNS IN A
021A  FE FFDA A          LDX    ADR
021D  BD 03AF I          JSR    SETOFF        STORE IN MEM(X), VERIFY
0220  08                 INX
0221  FF FFDA A          STX    ADR
0224  7A FFE3 A          DEC    COUNT         DOES COUNT=0?
0227  2E EE              BGT    LDR10         NO, CONTINUE LOOP
                  *END UNTIL LOOP
0229  20 04              BRA    LHF9
022B  81 39       LHF4   CMP A  #'9           EOF RECORD ?
022D  26 13              BNE    BADTAP        ILLEGAL RECORD TYPE
                  *
022F  BD 011C I   LHF9   JSR    CHEKSM        CHECK CKSM
0232  B6 FFE2 A          LDA A  RECTYP        GET RECORD TYPE
0235  81 39              CMP A  #'9           EOF RECORD ?
0237  26 A4              BNE    LHF2          NO. CONTINUE LOOP
                  *
                  *END OF OUTER LOOP
                  *
0239  BD 039B I          JSR    RDROFF
023C  CE 026F I          LDX    #MEOF         PRINT "EOF"
023F  7E 0080 I          JMP    MSGMON        AND RETURN TO MONITR LOOP
                  *
0242  BD 039B I   BADTAP JSR    RDROFF
                  *
0245  CE 0281 I          LDX    #MTAPER       PRINT "TAPE ERR"
0248                     SUBR   PMSG
                  *
                  * ACCEPT NO COMMANDS UNTIL USER PRESSES ESC
                  *
024A  7C FFE9 A          INC    ECHO          SET ECHO
                  *
024D  BD 0400 I   BT1    JSR    WAITTY        ESC CAUSES ABORT
0250  20 FB              BRA    BT1
                  *
                  ***********************************************
                  *
                  * FIND S
0258  81 53              CMP A  #'S           CHAR = S
025A  26 F9              BNE    FS10          NO
                  *END LOOP
025C  39                 RTS
                  *
                  * MESSAGES
                  *
025D       4241   MBADR  CHAR   /BAD ADR/
025F       4420
0261       4144
0263       52
0264       04            BYTE   4
0265       5241   MRNGER CHAR   /RANGE ERR/
0267       4E47
0269       4520
026B       4552
026D       52
026E       04            BYTE   4
026F       454F   MEOF   CHAR   /EOF/
0271       46
0272       04            BYTE   4
0273       3F3F   MQUES  CHAR   /????/
0275       3F3F
0277       04            BYTE   4
0278       434B   MCSER  CHAR   /CKSM ERR/
027A       5340
027C       2045
027E       5252
0280       04            BYTE   4
0281       5441   MTAPER CHAR   /TAPE ERR/
0283       5045
0285       2045
0287       5252
0289       04            BYTE   4
028A       5339   MPEOF  CHAR   /S9030000FC/
028C       3033
```

Advanced Microcomputer Products Presents

CONTINENTAL SPECIALTIES CORPORATION
CSC Prototype Design Equipment

QT SOCKETS AND BUS STRIPS



PROTO-BOARD® UNITS

PROTO-BOARD 6 KIT

Now in stock for Microprocessor Prototyping
NEW SOCKETS & BUS STRIPS
Experimenter 600 . . . . $10.95
.6 spacing for microprocessor applications
Experimentor 300 . . . . $ 9.95
.3 spacing for microprocessor applications

PROTO-BOARD 100 KIT
PROTO-BOARD 101
PROTO-BOARD 102
PROTO-BOARD 103
PROTO-BOARD 104
PROTO-BOARD 203
PROTO-BOARD 203A

LOGIC PROBE LP-1

WE ARE A FULL LINE CSC DISTRIBUTOR. WE CAN SUPPLY ANY PRODUCT OUT OF THE CSC CATALOGUE. IF YOU NEED THEIR LATEST CATALOGUE WRITE AND WE'LL SEND IT ALONG

8K RAM BOARD $198.00

ADVANCED ★ MICROCOMPUTER PRODUCTS

BUILD YOUR OWN COMPLETE COMPUTER SYSTEM

★ AMI EVK 99 6800 BASED MICROCOMPUTER

Ask the Home Brew Computer Club in Santa Clara, California why they choose the EVK 99 Microcomputer. They now have 85 systems. The reasons are clear: Performance, versatility and cost effectiveness. Photo is of expanded EVK 99 Board

Photo of expanded EVK 99
$159.95

1. EVK 99 consists of the following only
   Qty (1) PCB 10½ x 12
   (1) 6800 8 bit CPU
   (1) 6850 ACIA (Asynchronous Communication Interface Adapter)
   (1) 6820PIA (Peripheral Interface Adapter)
   (2) 6830 ROM Subroutine Program Library (1) 6831 16K ROM
   (4) 68101 RAM
   (1) Operating Manual & instructions
   To implement fully expanded system additional IC's are required (See Photo)
2. UNIVERSAL KLUGE BOARD $98.00
   Completely compatible to the EVK system. This board allows the user to add any additional parts he desires. It has 5400 holes, holds 94 16 Pin IC's or others on 2, 4, 6, 9" or greater centers. 2 50 pin, 1 20 pin flatcable, 1 25 pin RS232 connector
3. 16K BYTE RAM BOARD $77.95
   16K x 8 or 8K x 16 organization. Uses 2102's or 21L02's and adapts to slow memory
4. 6 SLOT MOTHER BOARD $29.95
   Connectors mount on .75" centers. May be plugged together for expanded applications
5. EXTENDER BOARD $46.50
   For use with back plane above in card cage configuration
   New boards are now in development including cassette I/O and others. If you want more information drop us a line.

★ MICROPROCESSOR PRODUCTS

8080A/9080A	$24.95	S6860		$15.95		21L02/91L02	1.99
AMI S6800	24.95	S6820		11.00		2102/9102	1.89
AM2901	29.95	S6810 1		6.25		9111/2111	4.75
GI CP1600	69.95	S6850		10.95		9101/2101	5.25
Z80	69.95	S2350		12.95		9112/2112	2.95
6502	29.95	S1883		5.90		7489	1.88
AM2902PC	$3.95	S6834 1		22.95		4104/4604	12.95
AM2905PC	10.95	S6800	Programming Manual	15.00		4402A	12.95
AM2906PC	11.95	S6800	Hardware Manual	15.00		4200	12.95
AM2907PC	10.75		Tiny Basic	22.95		9131APC	14.95
AM2909PC	24.95		On 6834 1 Rom	175.00		9141APC	14.95
AM2911PC	18.95	8212	$4.75	AM2533	7.50	9050/TMS4050	12.95
AM2918	6.75	8224	6.25	AM2401	16.50	9060/TMS4060	12.95
AM29720PC	7.95	8216	4.95	AM1402	6.95	P3101	3.25
AM29721PC	7.95	8226	5.95	AM1403	5.95	5101	11.95
AY 51013A	5.50	8228	9.90	AM1404	6.95	1702A	10.90
AM2812	14.95	9551	19.95	MM5008H	2.25	2708	68.00*
AM2841	12.00	9555	19.95	MM5016H	4.95	5204	23.50
AM3341	9.95			AM3347	6.40		

TV GAME KITS

Qty (1) GI AY38500 1 TV Game Chip — ONLY $29.95

TV KIT NO. 1 — ONLY $34.95
Includes AY38500 1 TV GameChip, PC Board for video composite output plus instructions. Add $5.95 for 2MHz crystal

TV KIT NO. 2 — ONLY $49.95
Includes AY38500 1 TV GameChip, PC Board, Switches, pots, speaker, crystal, XFMR, Bridge and all parts required for video composite. Works super with terminal. Includes instructions and data sheets

SPECIAL
BUY 10 OF THE FOLLOWING AND YOUR PRICE IS. . .
AMD 8080A $19.95
AMD 9111 .... 4.10
AMD 9112 .... 2.30

RF MODULATOR MODULE

To convert your video composite to RF try UM 1021 Channel 3/4 VHF Modulator $8.95
● DC Switchable VSWR = 2 max VCC = 6.5 volts
● DC Coupled modulation input with positive transfer characteristic. Designed for use with TV Games

SPECIALS

75452	10/2.50
8T26B	2/5.00
74LS138	1.49
8097	.99
74166	.80
74161	.49
741HC	10/2.50
MH0026CN	3.95
1488	1.29
1489	1.29

★ NEW PRODUCTS

NEW FROM NEC Microcomputer! FIRST TIME OFFERED

D372 Floppy Disc Controller Chip	$75.00
D371 Mag. Tape Cassette Controller Chip	65.00
Application Notes (26 pages)	6.95
GI AY3550 4½ Digit DVM Chip	29.95
GI AY3600 Keyboard Encoder	9.95
Intersil 8038CC	4.95
AY3507	9.95
RO 32513 Char Gen	9.95

★ IC MARKET PLACE ★

★ LINEAR

LM105	3.50	LM320K 5.2	1.45	LM381N	1.79	LM1414N	1.75	75324	3.50
LM106H	2.50	LM320K 12	1.45	LM382N	1.79	LM1458CN	.65	SG4501T	2.40
LM300H	.80	LM320K 15	1.45	LM555CN	.58	LM1496N	.95	SG4501N	2.40
LM301H	.45	LM320T 5	1.75	LM556CN	1.30	LM2111N	1.95	SG3524	8.95
LM301CN	.45	LM320T 5.2	1.75	LM567CN	1.45	LM2901N	1.95	RC4194T	6.95
LM302H	1.25	LM320T 8	1.75	LM565N	1.95	LM3065N	.69	RC4195T	6.95
LM304H	1.20	LM320T 12	1.75	LM565H	1.95	LM3900N	.49	RC1458CN	.69
LM305H	.85	LM320T 15	1.75	LM703CN	.45	LM3909	1.10	RC1556CN	.99
LM305AH	1.25	LM323K 5	9.95	LM709H	.45	LM7520	2.25	RC3403AO	2.30
LM306N	.95	LM324M	1.65	LM709N	.40	LM7524N	1.75	RC4131CN	.99
LM 307CN	.35	LM3 N	1.55	LM710N	.70	LM7525N	1.90	RC4136D	1.90
LM307H	.35	LM340K5	1.95	LM711N	.39	LM7534N	2.20	RC4558CN	.99
LM308H	1.00	LM340K 8	1.95	LM711H	.39	LM7535N	1.25	RC4151CN	6.50
LM308CN	1.00	LM340K 12	1.95	LM723N	.55				
LM309H	1.10	LM340K 15	1.95	LM723H	.55				
LM309K	1.75	LM340K 18	1.95	LM733N	1.00				
LM310CN	1.15	LM340K 24	1.95	LM733CH	1.45				
LM310H	1.15	LM351CN	.65	LM739N	1.29				
LM311H	.90	LM370N	1.65	LM741CH	.35				
LM311CN	.49	LM371H	1.65	LM741CN 14	.39				
LM312H	1.95	LM373N	3.25	LM741CD	.49				
LM318H	1.50	LM377N	4.00	LM747H	.75				
LM318CN	1.50	LM380N	1.39	LM747N	.75				
LM319N	1.30	LM380CN	1.05	LM748H	.39				
LM320K 5	1.45			LM748N	.39				

★ CMOS

CD4000	.25	CD4026	1.95	CD4075	.35
CD4001	.25	CD4027	.85	CD4078	.35
CD4002	.25	CD4028	1.50	CD4081	.35
CD4006	1.75	CD402	1.99	CD4082	.35
CD4007	.25	CD4030	.60	CD4510	1.95
CD4008	1.95	CD4033	2.49	CD4511	1.40
CD4009	.48	CD4034	4.95	CD4516	1.95
CD4010	.48	CD4035	1.79	CD4518	2.45
CD4011	.28	CD4040	2.10	CD4520	2.49
CD4012	.28	CD4042	1.45	CD4522	2.56
CD4013	.60	CD4043	1.45	CD4526	2.56
CD4014	1.95	CD4044	1.95	CD4527	2.70
CD4015	1.95	CD4049	.61	CD4528	1.50
CD4016	.59	CD4050	.61	CD4531	1.99
CD4017	1.25	CD4051	2.25	CD4543	2.25
CD4018	2.39	CD4066	1.20	CD4555	1.75
CD4019	1.95	CD4068	.35	CD4556	1.75
CD4020	1.35	CD4069	.35	CD4581	4.95
CD4021	1.85	CD4070	.99	CD4582	1.95
CD4023	1.29	CD4071	.35	CD4585	2.45
CD4024	1.29	CD4072	.35	Also	
CD4025	.29	CD4073	.35	5101	$11.95

★ 7400 TTL

7400N	.18	7445N	.99	74122N	.48	74185N	$2.10
7401N	.18	7446N	.99	74123N	.80	74190N	$1.30
7402N	.18	7447N	.99	74125N	.79	74191N	$1.30
7403N	.18	7448N	.99	74126N	.79	74192N	$1.10
7404N	.19	7450N	.26	74132N	.70	74193N	$1.10
7405N	.20	7451N	.26	74141N	.88	74194N	.95
7406N	.25	7453N	.26	74145N	.70	74195N	.95
7407N	.25	7454N	.26	74147N	$1.59	74196N	.95
7408N	.21	7460N	.45	74148N	$1.29	74197N	$1.45
7409N	.21	7470N	.45	74150N	$1.15	74198N	$1.85
7410N	.21	7472N	.30	74151N	.98	74199N	$1.45
7411N	.21	7473N	.30	74153N	.98	74251N	$1.45
7412N	.21	7474N	.30	74154N	$1.15	74279N	$1.10
7413N	.45	7475N	.69	74156N	.95	8093N	.49
7414N	.69	7476N	.47	74158N	.95	8094N	.49
7416N	.35	7480N	.70	74157N	.95	8095N	.99
7417N	.35	7482N	.95	74160N	.99	8096N	.99
7420N	.19	7483N	.95	74161N	.99	8097N	.99
7421N	.35	7485N	.89	74162N	.99	8098N	.99
7423N	.35	7486N	.28	74163N	.99	75150N	$1.70
7425N	.35	7489N	$1.88	74164N	.99	75450N	$1.10
7426N	.35	7490N	.60	74165N	.99	75451N	.59
7427N	.35	7491N	.70	74166N	$1.10	75452N	.59
7428N	.78	7492N	.59	74170N	$4.20	75453N	.59
7430N	.21	7493N	.79	74172N	$18.75	75454N	.59
7432N	.31	7494N	.79	74173N	$1.40	75491N	.70
7433N	.35	7495N	.79	74174N	$1.25	75492N	.70
7437N	.35	7496N	.79	74175N	$1.25	75493N	$1.10
7438N	.35	7497N	$3.75	74176N	.79	75494N	$1.10
7440N	.21	74100N	$1.10	74177N	.79		
7441N	$1.05	74104N	.53	74180N	.79		
7442N	.69	74106N	.53	74181N	$2.75		
7443N	.69	74107N	.33	74182N	.79		
7444N	$1.50	74109N	.33	74184N	$2.10		
		74121N	.35				

100 PIECES MIX TAKE 10% DISCOUNT
1000 PIECES MIX TAKE 15% DISCOUNT

PLESSEY SEMICONDUCTORS

RADIO COMMUNICATION

SL610C	1.95	SL624C	2.75
SL611C	1.95	SL630C	1.95
SL612C	1.95	SL640C	1.95
SL613C	2.40	SL641C	6.95
SL620C	2.55	SL645C	5.95
SL621C	2.55	SL650C	5.75
SL622C	2.95	SL651C	4.95
SL623C	2.95	SL680C	3.75

Application Manual for SL600 Circuits 1.95
Plessey Semiconductor Catalogue 3.95

DIVIDERS

SP8600B	6.95	SP8620B	12.50	SP8640B	6.20
SP8601B	4.25	SP8621B	8.50	PS8641B	7.50
SP8602B	16.00	SP8622B	6.50	SP8642B	17.50
SP8603B	14.50	SP8630B	20.00	SP8643B	20.00
SP8604B	8.25	SP8631B	12.00	SP8646B	6.50
SP8607B	17.00	SP8632B	10.00	SP8647B	8.00
SP8613B	17.50	SP8634B	35.00	SP8650B	25.00
SP8614B	18.00	SP8635B	27.50	SP8651B	18.50
SP8615B	20.00	SP8638B	22.50	SP8652B	12.00
SP8616B	22.00	SP8637B	18.00		

ALTAIR /IMSAI COMPATIBLE 8K MEMORY

Our new brain board comes complete with 21L02-1 memory chips (500ns) and sockets
All address control and data out lines fully buffered.
THE SAME 8K BOARD OFFERED TO THE SCCS GROUP PURCHASE NOW OFFERED TO YOU AT THE SAME LOW LOW PRICE!
S198.00 COMPLETE
If you desire 91L02APC Please add $9.95

TMS 4060 RAM	$4.95
NEC 4096 RAM	4.25
1602B UART	5.50
1502F FIFO	12.95
34702 BAUD GEN	14.95
NEC 5101C-E CMOS RAM	6.95
RO32513 CHAR GEN Lower Case	14.95
WD 1671B ASTROS	29.95
WD 1941 DUAL BAUD GEN	9.95
WD 1771 FLOPPY DISC CONT.	74.95

SPECIAL

100 22PIN 4096 OYN RAM	3.25
100-22 PIN 4096 DYN RAM	3.25
100-16 PIN 4096 DYN RAM	3.45

6800 STARTER SET

Only 68 Bucks $6800
You Get (1) 6800 8 bit CPU
(1) 6820PIA
(1) 6850ACIA
(16) 2102RAMs
Add $15.00 for AMI 6800 Programming Manual

8080A STARTER SET

Only $80.00
You Get (1) 8080AB bit CPU
(1) 8212 I/O Port
(1) 8224Clock
(1) 8228 System Controller
(24) 2102 RAMs
Add $5.95 for MCS 8 8080 Users Manual

FCN GEN KIT	$18.50
63 Key Keyboard	19.95
19 Key Keyboard	11.95

Also . . . we stock entire continental specialties line new sockets and bus strips for microprocessor design

Experimenter 600 (.6 spacing)	$10.95
Experimenter 300 (.3 spacing)	9.95

All shipments first class or UPS in U.S. or add 5% for shipping and handling. Minimum order $10.00. California Residents add 6% tax. Foreign add 8%.

Z-80 $69.95 4K STATIC RAM ★ 2708 $49.95

TRANSISTORS

P/N	DESCRIPTION	PKG	UNIT PRICE	10	100	1000
2N918	NPN High Frequency	TO-18	.24	.17	.16	.15
2N930	NPN Low Level Low Noise	TO-18	.24	.17	.16	.16
2N2222A	NPN General Purpose	TO-92	.19	.165	.155	.135
2N2219A	NPN General Purpose	TO-5	.23	.17	.16	.15
2N2368A	NPN High Speed Switch	TO-92	.23	.165	.155	.135
2N2905A	PNP Small Sig. General Purpose	TO-5	.22	.20	.18	.17
2N2907A	PNP General Purpose	TO-18	.21	.175	.165	.155
2N3392	NPN Small Signal G.P.	TO-92	.19	.15	.14	.13
2N3414	NPN Gen. Purpose Switch	TO-92	.20	.19	.165	.18
2N3563	NPN Small Signal HF	TO-92	.24	.165	.14	.13
2N3565	NPN Low Level High Gain	TO-92	.24	.165	.14	.13
2N3638	PNP Sm. Sig. GP Amp	TO-106	.24	.165	.14	.13
2N3640	PNP Hi Speed Sat. Switch	TO-92	.24	.165	.14	.13
2N3646	NPN Hi Speed Sat. Switch	TO-92	.24	.165	.14	.13
2N3904	NPN Sm. Sig. GP Amp	TO-92	.23	.15	.14	.13
2N3906	PNP Sm. Sig. GP Amp	TO-92	.23	.15	.14	.13
2N4092	Nch. Fet. VP 20 8V 40	TO-18	.48	.43	.41	.40
2N4093	Nch. Fet. VP 20 8V 50	TO-18	.50	.45	.43	.42
2N4123	NPN Sm. Sig. GP Amp	TO-92	.23	.15	.14	.13
2N4125	PNP Sm. Sig. GP Amp	TO-92	.23	.15	.14	.13
2N4400	NPN GP Switch	TO-92	.23	.155	.145	.135
MPSA13	NPN Darlington (hfe 5000)	TO-92	.28	.27	.24	.225
2N3055	NPN Power 15A 115 Watt	TO-3	.95	.90	.78	.72
MJ3055	Plastic (Equivalent to MJE3055)	TO-220	.95	.90	.80	.77
2N3725	Core Driver	TO-5	.65	.50	.35	.78
FPQ3725CN	Quad Core Driver	DIP	2.50	1.95	1.75	1.40

BRIDGES ★

			1-99	100-499	500-999
W02M	75A	200V	.74	.46	.35
W005M	75A	50V	.70	.43	.34
W06M	75A	600V	.82	.52	.46
W08M	75A	800V	.88	.56	.47
W10M	75A	1000V	1.05	.66	.49
KBPC25-02	25A	200V	2.50	2.25	1.95
KBPC25-04	25A	400V	2.60	2.30	1.99
KBPC25-06	25A	600V	2.75	2.35	2.10

$1.00 Minimum per line item.

CRYSTALS

Microprocessors Timebases
TV Games Precision Crystals

Frequency	Case	Price	Frequency	Case	Price
1.000 MHz	HC6/HC33	4.95	13.000 MHz	HC18	4.95
2.000 MHz	HC6/HC33	5.85	14.31818 MHz	HC18	4.95
2.097152MHz	HC6	6.95	18.000 MHz	HC18	4.95
4.000 MHz	HC18	4.95	20.000 MHz	HC18	4.95
5.000 MHz	HC18	4.95	32.000 MHz	HC18	4.95
6.000 MHz	HC18	4.95	100Kc	HC13	12.95
10.000 MHz	HC18	4.95	3579.45 Kc	HC18	4.95

★ Exar

★ FUNCTION GENERATOR KIT ★
The Function Generator Kit features sine, triangle and square wave; THD 0.5% typ., AM/FM capability.
XR-2206KA includes monolithic function generator IC, PC board, and assembly instruction manual 18.95

★ TIMING CIRCUITS ★

XR555CP	Monolithic Timer	.90
XR556CP	Dual 555 Timer	1.75
XR2556CP	Dual Timing Circuit	2.95
XR2240CP	Programmable Timer	4.20
XR320P	Timer	1.40

★ COMMUNICATION CIRCUITS ★

XR1488	Quad RS232 Driver	1.45
XR148	Quad RS232 Receiver	1.45
XR2208	Four Quadrant Multiplier	4.75

★ PHASE-LOCKED LOOPS ★

XR2211CP	FSK Demod/Tone Decoder	5.95
XR567CP	Tone Decoder	1.80
XR25567CP	Tone Decoder	1.80

★ STEREO DECODERS ★

XR1310P	FM Stereo Demodulator	3.00
XR1310EP	FM Stereo Demodulator	3.00
XR1800	Stereo Demodulator	3.00

★ FUNCTION GENERATORS ★

XR2206CP	Monolithic Function Generator	5.25
XR2207CP	Current Controlled Oscillator	3.75

★ OPERATIONAL AMPLIFIERS ★

XR742CP	Zero Crossing AC Trigger	3.00
XR4558CP	Dual Hi Gain Op Amp	1.20
XR4136CP	Quad Op Amp	1.75
XR4202CP	Programmable Quad Op Amp	4.25

2708 PRIME ONLY $49.95

Build Your Own **MONITOR**
Now in Stock — NEW

M3000 - 100 12-inch display module	$229.00
M2000 - 155 9-inch display module	$208.00

*Add $10.00 Shipping

MOTOROLA Exorciser, MEK6800DI and MEK6800D2 Compatible Modules

9601 16 Slot mother board - Inc. Conn.	$175.00
9602 16 Slot card cage 19" rack mount	$ 72.00
9610 Utility Prototyping Board	$ 36.00
9615 4K Eprom Module (1702A)	$350.00
9620 Parallel I/O Module	$375.00
9626 8K Static Ram Module	$350.00
9650 8 Channel Duplex Ansyn I/O	$390.00

All assembled and tested. Not Kits

★ MISCELLANY

RIBBON CABLE
8 Conductor
1-9 ft. 10-49 ft. 50 ft up
.35/ft .25 .23

ELCO VARIPAK 2 H
2-Slot cardholders
4/$1.00

3-5V BUZZER $.95

LED'S
.185" Dia (Red) 5/$1.00
.185" Dia (Green) 4/$1.00
.185" Dia (Yellow) 4/$1.00

KEYBOARD

63 key array, terminal keyboard, unencoded with high quality switches. Black w/white legends. Special Offer: Buy now at $19.95 and we will give you a free calculator type keyboard.

AY5237 Keyboard Encoder with keyboard Only $14.95

TOUCHTONE KEYBOARD FOR TELEPHONE
Use with MK5085 N integrated tone dialer & 3.579 Hz crystal. Crystal available for $4.95
Only $9.95

NOW!
A 19 key Number Keyboard compatible with our 63 Key Keyboard. Same manufacture and design. Keys for 0 to 9 and 110,300, FDX/HDX, minus Auto LF, Line Loc, Alt Char set and Char set lock.
ONLY $11.95 FIRST TIME OFFERED (Identical to our 63 Key as pictured except 19 keys.)

★ CLOCK CHIPS

MM5311	24 Pin BCD Outputs 6 Digits	$4.50
MM5313	28 Pin 6 Digits	4.00
MM5314	24 Pin 6 Digits	3.50
MM5316	40 Pin Alarm Ledout	4.50
MM5375	Clock Chip	3.50

★ IC SOCKETS

Low Profile Tinplate
8 Pin .17 24 Pin .41
14 Pin .20 28 Pin .49
16 Pin .22 40 Pin .63
18 Pin .31

★ RESISTORS

Carbon Resistors 1/4 Watt ±5%
Most Standard Resistance Values.
NOTE: Multiples of 5 pcs per value minimum

Qty. (each value)	Price
5-20	.05 each
25-50	.045 each
55-100	.04 each
105-500	.03 each
505 up	.025 each

★ VARIABLE RESISTORS

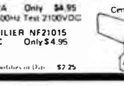
Centralab RN4NAYSD503A (HMP50K) $1.95
RN4 NAYSD503A (53C350KS) $1.95

★ CORCOM EMI FILTER

● CORCOM 2B4 2A Only $4.95
115/230V 50-400Hz Test 2100VDC
● CORNELL DUBILIER NF21015
2x4A250VAC Only $4.95
● DIP SWITCH
8 SPST Slide Switches in Dip $2.25

ROTARY SWITCH

Centralab PA 2001 $1.95



C&K V 25 Bat handle toggle switch $1.50

ADVANCED ★ MICROCOMPUTER PRODUCTS
P. O. BOX 17329
IRVINE, CALIFORNIA 92713
PHONE (714) 968-3655

★ ORDER YOUR NEW CATALOG NOW!

```
028E      3030
0290      3030
0292      4643
0294      04                   BYTE    4
0295      0D0A      MCRLFS     BYTE    CR,LF,0,0,0,0,'S,'1,4
0297      0000
0299      0000
029B      5331
029D      04
                    *
                    ***************************************************
                    *
                    * NEXT 2 DIGITS--
                    *   READ NEXT 2 CHAR FROM TTY TAPE AND CONVERT
                    *   TO HEX NUMBER IN A REG.  UPDATE CKSM.
                    *   RETURN UPDATED CKSM IN B REG.
                    *
                    ***************************************************
            029E I  NEXT2D     EQU     *
029E  BD 0400 I                JSR     WAITTY          GET CHAR
02A1  16                       TAB                     SAVE CHAR IN A
02A2  BD 0400 I                JSR     WAITTY
                    *
                    * SET UP PARAMS FOR CONVERSION ROUTINE.
                    * PUSH ASCII CHARS INTO STACK. POINT X AT STACK.
                    * SET A=TYPE OF CONVERSION AND B=# OF CHARS TO CONVERT.
                    *
02A5  36                       PSH A
02A6  37                       PSH B
02A7  30                       TSX
02A8  C6 02                    LDA B   #2
02AA                           SUBR    CONVHB          CONVERT FROM ASCII TO BINARY
02AC  24 94                    BCC     BADTAP          IF NON-HEX CHAR, ABORT
                    *
02AE  17                       TBA                     UPDATE CKSM
02AF  FB FFE4 A                ADD B   CKSM
02B2  F7 FFE4 A                STA B   CKSM
02B5  31                       INS                     RESTORE STACK PTR
02B6  31                       INS
02B7  39                       RTS
                    *
                    ***************************************************
                    *
                    * NEXT ADR(BUFPTR,ADR)
                    *
                    *    SET ADR=0 OR NEXT NUMBER STRING STARTING
                    *    AT BUFPTR
                    *    LEAVES BUFPTR AT CR,DELIMITER,OR FIRST
                    *    CHAR BETWEEN G - Z .
                    *    LEAVES (A)= LAST CHAR SCANNED.
                    *    LEAVES (B)= LS BYTE OF ADR
                    *
                    *    RETURNS:  CC= Z  FOR  NO PARAMETER
                    *              ABORTS IF NON-HEX PARAMETER
                    *
                    *
                    ***************************************************
            02B8 I  NXTADR     EQU     *
02B8  7F FFDA A                CLR     ADR             ADR:= 0
02BB  7F FFDB A                CLR     ADR+1
02BE  BD 0385 I                JSR     PXISTS          IS THERE A PARAMETER?
02C1  26 01                    BNE     NA1             YES
02C3  39                       RTS                     RETURN W/NO PARAM  CC=Z
                    *
                    *
                    * SET UP PARAMS FOR ASCII TO HEX CONVERSION
                    *
02C4  C6 47         NA1        LDA B   #71             MAX. CHARS TO SCAN
02C6                           SUBR    CONVHB
02C8  FF FFE0 A                STX     BUFPTR
02CB  B7 FFDA A                STA A   ADR             SAVE RESULT
02CE  F7 FFDB A                STA B   ADR+1
02D1  A6 00                    LDA A   0,X             CHECK TERMINATOR
02D3                           SUBR    ALPNUM          IS CHAR ALPHA?
02D5  25 01                    BCS     NA3             YES
02D7  39                       RTS
                    *
02D8  7E 00AD I     NA3        JMP     ABORT           NO
                    *
                    *
```



```
                    ***************************************************
                    *
                    * OUTCH = PRINT CHAR IN A
                    * OUTCHX = PRINT CHAR AT MEM(X)
                    *    IF CHAR = 'CR', FOLLOW WITH LF & 4 NULLS
                    *
                    ***************************************************
                    *
02DB  A6 00         OUTCHX     LDA A   0,X             ENTRY 1
                    *
            02DD I  OUTCH      EQU     *
                    * FIRST CHECK FOR ESC
02DD  37                       PSH B
02DE  F6 FBCE A                LDA B   ACIAS           ACIA INPUT STATUS
02E1  57                       ASR B                   C:=RDRF
02E2  24 0A                    BCC     OC10            NO INPUT
02E4  F6 FBCF A                LDA B   ACIAD           READ ACIA
02E7  C1 1B                    CMP B   #ESC
02E9  26 03                    BNE     OC10            NOT ESC
02EB  7E 00AD I                JMP     ABORT
                    *
02EE                OC10       SUBR    PUTA            PRINT CHAR
02F0  81 0D                    CMP A   #CR
02F2  26 0E                    BNE     OC20            NOT CR. RETURN
                    *
02F4  86 0A                    LDA A   #LF             PRINT LF
02F6                           SUBR    PUTA
02F8  4F                       CLR A                   PRINT 4 NULLS
02F9  C6 04                    LDA B   #4
                    * BEGIN LOOP
02FB                OCLOOP     SUBR    PUTA
02FD  5A                       DEC B
02FE  26 FB                    BNE     OCLOOP
                    * END LOOP
0300  86 0D                    LDA A   #CR             RESTORE A
                    *
0302  33            OC20       PUL B
0303  39                       RTS
                    *
                    ***************************************************
                    *
                    * PRINT CR,LF,NULL
                    *
                    ***************************************************
0304  86 0D         PCRLF      LDA A   #CR
0306  20 D5                    BRA     OUTCH           OUTCH PRINTS LF AFTER CR
                    *
                    *
                    ***************************************************
                    *
                    * PUNCH  ADDL,ADDH
                    *     PUNCH MEMORY CONTENTS BETWEEN ADDL & ADDH
                    *     IN HEX FORMAT
                    *
                    ***************************************************
0308  BD 0160 I     PUNCH      JSR     GETRNG          READ ADDL & ADDH+1
030B  CE 0000 A                LDX     #0
030E  FF FFD8 A                STX     OFFSET
0311  8D A5                    BSR     NXTADR          ANY OFFSET?
0313  27 06                    BEQ     PHF15           NO.
0315  FE FFDA A                LDX     ADR             YES.
0318  FF FFD8 A                STX     OFFSET
                    *
                    * PUNCH DATA RECORDS UNTIL ADDL = ADDH
                    *
            031B I  PHF15      EQU     *
                    * BEGIN LOOP
                    *
                    * CALCULATE DATA LENGTH = MIN(30, ADDH+1-ADR)
                    *
031B  F6 FFDF A     PHF20      LDA B   ADDH+1          B:=ADDH-ADDL
031E  F0 FFDD A                SUB B   ADDL+1
0321  B6 FFDE A                LDA A   ADDH
0324  B2 FFDC A                SBC A   ADDL
0327  26 04                    BNE     PUND10          DIFF .GT. 256
0329  C1 1E                    CMP B   #30             LS BYTE .GT. 30?
032B  23 02                    BLS     PUND20
                    *
032D  C6 1E         PUND10     LDA B   #30             DIFF .GT.30
                    *
032F  5C            PUND20     INC B                   COUNT:=COUNT+3
0330  5C                       INC B                   ...INCLUDES ADDR & CKSM
0331  5C                       INC B
0332  F7 FFE3 A                STA B   COUNT
0335  CE 0295 I                LDX     #MCRLFS
0338                           SUBR    PMSG
033A  5F                       CLR B                   B HOLDS CKSM
033B  CE FFE3 A                LDX     #COUNT          PUNCH COUNT
033E  8D 34                    BSR     PUNBYT
0340  37                       PSH B
0341  FE FFDC A                LDX     ADDL            COMPUTE OFFSET ADDRESS
0344  B6 FFD8 A                LDA A   OFFSET
0347  F6 FFD9 A                LDA B   OFFSET+1
034A                           SUBR    ADDABX
034C  FF FFDA A                STX     ADR             PUNCH FROM ADR
034F  CE FFDA A                LDX     #ADR
0352  33                       PUL B
                    *
0353  8D 1F                    BSR     PUNBYT          (INCREMENTS X)
0355  8D 1D                    BSR     PUNBYT
0357  FE FFDC A                LDX     ADDL            RESTORE X
                    *
                    * PUNCH BYTES FROM MEMORY UNTIL COUNT IS EXHAUSTED
                    *
                    * BEGIN LOOP
035A  8D 18         PREC10     BSR     PUNBYT          (CC=0 IF COUNT=0)
035C  2E FC                    BGT     PREC10
                    * END LOOP
035E  FF FFDC A                STX     ADDL            SAVE X
0361  CE FFE4 A                LDX     #CKSM           PUNCH CKSM
0364  53                       COM B
0365  E7 00                    STA B   0,X             CKSM:=B
0367  8D 0B                    BSR     PUNBYT
0369  FE FFDC A                LDX     ADDL
036C  BC FFDE A                CPX     ADDH
036F  26 AA                    BNE     PHF20
                    * END LOOP
0371  7E 0041 I                JMP     MONEN1
                    *
                    ***************************************************
                    *
                    * PUNBYT (MEM(X),COUNT,CKSM)
                    *    PUNCH BYTE AT MEM(X) AND ADJUST COUNT AND CKSM.
                    *    CC=Z IF COUNT=0
                    *
                    ***************************************************
0374  EB 00         PUNBYT     ADD B   0,X             CKSM:=CKSM+MEM(X)
0376                           SUBR    P2HEX           PRINT MEM(X) AS 2 CHAR
0378  7A FFE3 A                DEC     COUNT
037B  39                       RTS
                    ***************************************************
                    *
                    * P4HEXS:  PRINT 2 BYTES AT X AS 4 HEX CHARS + 2 SPACES
                    *
                    ***************************************************
                    *
037C                P4HEXS     SUBR    P4HEX
037E  8D 00                    BSR     PSPACE
                    *
                    ***************************************************
                    *
                    * PSPACE---PRINT 1 BLANK
                    *
                    ***************************************************
0380  86 20         PSPACE     LDA A   #BLANK
0382                           SUBR    PUTA
0384  39                       RTS
                    *
                    *
                    ***************************************************
                    *
                    * PARAM EXISTS(BUFPTR)        (#BUFPTR) = BUFPTR
                    *                                   (X) = BUFPTR
                    * INC BUFPTR UNTIL CHAR = ALPHA OR CR
                    * LEAVE A = MEM(BUFPTR)
                    * SET Z IF NO PARAMETER EXISTS
                    *
                    ***************************************************
            0385 I  PXISTS     EQU     *               ENTRY FOR (#BUFPTR)=BUFPTR
```

```
0385  FE FFE0 A        LDX    BUFPTR
           0388 I PXISTX EQU   *              ENTRY FOR (X) = BUFPTR
                  *BEGIN LOOP
0388  A6 00       PX1    LDA A  0,X           IS CHAR  ALPHANUM ?
038A                     SUBR   ALPNUM
038C  25 07              BCS    PX2           YES, EXIT LOOP
038E  81 0D              CMP A  #CR           IS CHAR CR ?
0390  27 03              BEQ    PX2           YES, EXIT LOOP
0392  08                 INX                  MOVE TO NEXT CHAR
0393  20 F3              BRA    PX1
                  *END LOOP
0395  FF FFE0 A   PX2    STX    BUFPTR
0398  81 0D              CMP A  #CR           SET Z IF NO PARAMETER
039A  39                 RTS
                  ************************************************
                  *
                  * RDR OFF
                  *    TURNS TAPE RDR OFF:
                  *         ACIA RTS O/P HIGH
                  *         ACIA CHAR  $13  (DC3)
                  *
                  ************************************************
           039B I RDROFF EQU    *
039B  86 01              LDA A  #$01          RTS HIGH
039D  B7 FBCE A   ROF90  STA A  ACIAC         SET ACIA CONT REG
03A0  86 13              LDA A  #$13          SEND TTY RDR CONT CHAR
03A2                     SUBR   PUTA
03A4  39                 RTS
                  ************************************************
                  *
                  * RDR ON
                  *    TURNS TAPE READER ON
                  *         ACIA RTS O/P  LOW
                  *         ACIA CHAR  $11  (DC1)
                  *
                  ************************************************
           03A5 I RDRON  EQU    *
03A5  86 41              LDA A  #$41          RTS LOW
03A7  B7 FBCE A   RON90  STA A  ACIAC         SET ACIA CONT REG
03AA  86 11              LDA A  #$11          SEND TTY RDR CONT CHAR
03AC                     SUBR   PUTA
03AE  39                 RTS
                  *
                  ************************************************
                  *
                  * SETMEM(X)
                  * SETS MEM(X):=A AND VERIFY
                  *
                  ************************************************
                  *
           03AF I SETOFF EQU    *
03AF  36                 PSH A                FIRST CHECK RANGE:
03B0  B6 FFDC A          LDA A  ADDL          LOW LIMIT
03B3  F6 FFDD A          LDA B  ADDL+1
03B6                     SUBR   SUBXAB        16-BIT SUBTRACT
03B8  22 0A              BHI    SETOUT        TOO LOW
03BA  B6 FFDE A          LDA A  ADDH          HIGH LIMIT
03BD  F6 FFDF A          LDA B  ADDH+1
03C0                     SUBR   SUBXAB
03C2  24 07              BCC    SETPUL        OK
03C4  32          SETOUT PUL A                OUTSIDE RANGE LIMITS
03C5  86 FF              LDA A  #255          TYPE DELETE (RUBOUT)
03C7                     SUBR   PUTA          TO SIGNAL FACT TO USER
03C9  20 17              BRA    SETM1         OTHERWISE IGNORE STORE REQUEST
03CB  32          SETPUL PUL A
           03CC I SETMEM EQU    *
03CC  A7 00              STA A  0,X
03CE  A1 00              CMP A  0,X           VERIFY
03D0  27 10              BEQ    SETM1         ERROR ?
                  * VERIFY ERROR , PRINT ADR
03D2  FF FFDA A          STX    ADR           SET PARAM FOR P4HEX
03D5  CE FFDA A          LDX    #ADR
03D8  8D C1              BSR    RDROFF
03DA  8D 40              BSR    P4HEXS
03DC  CE 025D I   PBADR  LDX    #MBADR        PRINT 'BAD ADR'
03DF  7E 0080 I          JMP    MSGABT        PRINT MSG & ABORT
                  *
03E2  39          SETM1  RTS
                  ************************************************
                  *
                  * SM  ADR BYTE1,BYTE2,...
                  *
                  ************************************************
           03E3 I SM     EQU    *
03E3  BD 0288 I          JSR    NXTADR        ADR:= NEXT PARAM
                  *
03E6  FE FFDA A   SM5    LDX    ADR           SAVE ADR IN ADDL
03E9  FF FFDC A          STX    ADDL
                  *
                  * BEGIN WHILE LOOP
03EC  BD 0288 I   SM10   JSR    NXTADR        ADR:= NEXT PARAM
03EF  27 0C              BEQ    SM30          END OF LINE. EXIT LOOP.
03F1  FE FFDC A          LDX    ADDL          X:= ADD TO BE SET
03F4  17                 TBA                  A:=LS BYTE
03F5  8D D5              BSR    SETMEM        MEM(X):=A, VERIFY
03F7  08                 INX                  MOVE TO NEXT ADD
03F8  FF FFDC A          STX    ADDL
03FB  20 EF              BRA    SM10
                  * END OF LOOP
                  *
03FD  7E 0047 I   SM30   JMP    MONEND
                  ************************************************
                  *
                  * WAIT FOR TTY(CHAR,ECHO)  (#ECHO)=ECHO
                  *      RETURN NEXT TTY CHAR IN  A
                  *      IF (#ECHO) NOT 0 , ECHO CHAR
                  *
                  ************************************************
           0400 I WAITTY EQU    *
                  *LOOP UNTIL INPUT .NE. RUBOUT
0400              W10    SUBR   GETA          READ TTY
0402  81 1B              CMP A  #ESC          ESCAPE ?
0404  26 03              BNE    W20           NO
0406  7E 00AD I          JMP    ABORT         YES, ABORT
0409  81 7F       W20    CMP A  #RUBOUT       RUBOUT ?
040B  27 F3              BEQ    W10           YES CONTINUE LOOP
                  *END UNTIL LOOP
040D  7D FFE9 A          TST    ECHO
0410  27 03              BEQ    W30           NO ECHO
0412  BD 02D0 I          JSR    OUTCH         ECHO A
0415  39          W30    RTS
                         END
```

SYMBOL TABLE:

```
ABORT  00AD I    ACIAA  0005 I    ACIAC  FBCE A    ACIAD  FBCF A
ACIAI  FFF6 A    ACIAS  FBCE A    ADDABX 0008 A    ADDH   FFDE A
ADDL   FFDC A    ADR    FFDA A    ALPNUM 0013 A    AREG   FFED A
BADINP 00AD I*   BADTAP 0242 I    BASE   FF90 A    BLANK  0020 A
BOS    FF8F A    BREAK  0002 I    BREAK1 0087 I    BREG   FFEC A
BT1    024D I    BUF    FF90 A    BUFPTR FFE0 A    BURN   0001 R
CHEKSM 011C I    CKSM   FFE4 A    CONHB  0015 A    COUNT  FFE3 A
CR     000D A    CREG   FFEB A    CS1    0129 I    CTABLE 008C I
CTEND  00AD I    DL10   0082 I    DLOOP  007A I    DM     0136 I
DM10   0138 I    DM20   013E I    DM50   015A I    ECHO   FFE9 A
EOF    015D I    EOT    0004 A*   ESC    001B A    FINDS  0252 I
FS10   0255 I    G10    01AA I    GETA   0014 A    GETHG1 017E I
GETRG3 0184 I    GETRG4 0197 I    GETRNG 016D I    GO     019F I
IRQVEC FFF8 A    LAST   FFFF A*   LDR10  0217 I    LF     000A A
LHF2   01D0 I    LHF3   0213 I*   LHF4   0228 I    LHF9   022F I
LOAD   01AD I    LOOPST 0186 I    MBADR  025D I    MCRLFS 0295 I
MCSER  0278 I    MEOF   026F I    MONEN1 0041 I    MONEND 0047 I
MONENT 0041 I    MONITR 0047 I    MOVE   0002 R    MPEOF  028A I
MQUES  0273 I    MRNGER 0265 I    MSGABT 0080 I    MSGMON 0080 I
MTAPER 0281 I    NA1    02C4 I    NA3    02D8 I    NEXT20 029E I
NMIVEC FFFC A    NULL   0164 I    NULLS  0162 I*   NXTADR 0288 I
OC10   02EE I    OC20   0302 I    OCLOOP 02F8 I    OFFSET FFD8 A
OUTCH  02D0 I    OUTCHX 020B I*   P2HEX  000F A    P4HEX  0010 A
P4HEXS 037C I    PBADR  03DC I    PCRLF  0304 I    PHF15  0318 I
PHF20  0318 I    PINIT  0005 R    PMSG   0012 A    PR1    00F3 I
PR10   00F8 I    PR20   0102 I    PREC10 035A I    PREG   FFF0 A
PREGS  00EE I    PROMAD FFDB A    PRTXD  0009 A*   PSPACE 0380 I
PUNBYT 0374 I    PUNCH  0306 I    PUND10 0320 I    PUN020 032F I
PUTA   0011 A    PX1    0388 I    PX2    0395 I    PXISTS 0385 I
PXISTX 0388 I*   RDPRE  01E0 I    RDROFF 039B I    RDRON  03A5 I
READ   0003 R    RECTYP FFE2 A    RESTAR 010C I    RNGERR 0191 I
ROF90  0390 I*   RON90  03A7 I*   RSRSR  0000 R    RT10   005F I
RT20   0066 I    RT30   0069 I*   RT90   006C I*   RUBOUT 007F A
RUS10  0112 I    SAVESP FFE5 A*   SAVEX  FFE7 A*   SETM1  03E2 I
SETMEM 03CC I    SETOFF 03AF I    SETOUT 03C4 I    SETPUL 03C8 I
SM     03E3 I    SM10   03EC I    SM30   03FD I    SM5    03E6 I*
SREG   FFF2 A    START  0000 I*   START1 0007 I    SUBXAB 0008 A
SWI20  00CC I    SWI30  00D1 I    SW140  00D8 I    SW150  00D8 I
SWIHAN 008E I    SWIVEC FFFA A    TCOUNT FFEA A*   USWI   FFF4 A
VFY    0004 R    W10    0400 I    W20    0409 I    W30    0415 I
WAITTY 0400 I    XREG   FFEE A
```

```
CHECKSUM = 075E

LENGTH OF DSECT =       0 (0000)
LENGTH OF ISECT =    1046 (0416)

            NO ERRORS,  NO WARNINGS, THIS ASSEMBLY
```

```
PAGE   1   PROM      01/09/76  9:26  PROM BURNER ADDITION TO PHOTO

LOC   OBJECT  M  SOURCE STATEMENT

                  ************************************************************
                  *                                                          *
                          TITLE  PROM BURNER ADDITION TO PHOTO
                  *                                                          *
                  * PROM BURNER
                  *
                  * VERSION 2.0  01/08/76
                  *
                  * COPYRIGHT 1976 BY AMERICAN MICROSYSTEMS INC,
                  *
                  *
                  ************************************************************
                  *
                  *  ASSEMBLY OPTIONS
                  *
            0001 A  MOVER    EQU    1            0= MOVE ROUTINE EXCLUDED
            000A A  DELAY    EQU    10           POST PROGRAM DELAY, BEFORE VFY  (MS)
                             OPT    LSKP,LMAC
0000                         ISEC
0416                         ORG    $416
                             REF    MONENT,GETRNG,NXTADR,PXISTS,RNGERR,PBADR
                             REF    PCRLF,PSPACE,SETMEM,ABORT,MONITR
                             REF    PROMAD,ADR,ADDL,ADDH,COUNT
                             DEF    BURN,MOVE,READ,VFY,PINIT
                  *
                  *  PIA LOCATIONS:
                  *
            FBC0 A  PIA      EQU    H'FBC0
            0001 A  V50      EQU    H'FBC1-PIA
            0004 A  PROM     EQU    H'FBC4-PIA
                  *
                  *  STANDARD RAM BUFFER (DEFAULT)
                  *
            FC00 A  RAM      EQU    H'FC00
                  *
                  *  CHARACTER TYPING MACRO
                  *
                  TYPE       MACRO  CHAR
                           IF CHAR 0
                           LDAA #CHAR
                           IEND
                           CALL PRINTA
                            MEND
                  *
                  *  RSRSR CALL MACRO
                  *
                  CALL       MACRO  ITEM
                           SWI
                           BYTE ITEM
                            MEND
                  *
                  *  RSRSR CALL LOCATIONS
                  *
            0011 A  PRINTA   EQU    17
            0010 A  P4HEX    EQU    16
                  *
                  *  INITIALIZE PROM BURNER PIA'S
                  *
0416  CE FBC0 A  PINIT    LDX    #PIA
0419  86 38                LDAA   #B'00111000   TURN OFF 50V
041B  AA 01                ORAA   V50,X
041D  A7 01                STAA   V50,X
041F  86 3A                LDAA   #B'00111010
0421  A7 05                STAA   PROM+1,X      R/W TO READ
0423  A7 07                STAA   PROM+3,X      (HOPE NO DOUBLE-DRIVE HERE)
0425  6F 06                CLR    PROM+2,X      PROM DATA SET TO INPUTS
0427  6F 04                CLR    PROM,X
0429  63 04                COM    PROM,X        SELECT ADDRESS AS OUTPUTS
042B  86 3E                LDAA   #B'00111110
042D  A7 05                STAA   PROM+1,X      POINT TO ADDRESS OUTPUT REG.
042F  39                   RTS
                  *
                  *  TYPE A IN BINARY, ENCLOSED BY SPACES
                  *
0430  37         P8BIN    PSHB                  SAVE B
0431  36                  PSHA
0432  8D 0F               BSR    PSP           PRINT LEADING SPACE
0434  32                  PULA
0435  C6 08               LDAB   #8            8 DIGIT COUNTER
0437  49         :8       ROLA
0438  36                  PSHA
0439  86 18               LDAA   #24           (=1/2 ASCII "0")
043B  49                  ROLA
043C                      TYPE
              0  *        IF     0
                 *       LDAA #
                 *        IEND
043C             *        CALL   PRINTA
043C  3F         *        SWI
043D     11      *        BYTE   PRINTA
043E  32                  PULA
043F  5A                  DECB
0440  26 F5               BNE    :8
0442  33                  PULB
0443  7E 0007 R  PSP      JMP    PSPACE        PRINT ONE MORE SPACE
                  *
                  *  RAM/PROM ADDRESS SETUP & VALIDATION
                  *
0446  CE FC00 A  RASV     LDX    #RAM          INITITALIZE POINTERS TO DEFAULT RAM
0449  FF 000D R           STX    ADDL
044C  CE FE00 A           LDX    #RAM+512
044F  FF 000E R           STX    ADDH
0452  7F 000F R           CLR    COUNT         SET FULL PROM FLAG
0455  BD 0003 R           JSR    PXISTS        ...IF NO ADDRESS.
0458  27 06               BEQ    :A1
045A  BD 0001 R           JSR    GETRNG
045D  7C 000F R           INC    COUNT
0460  FE 000D R  :A1      LDX    ADDL          DEFAULT PROM ADDRESS
0463  FF 000B R           STX    PROMAD        IS SAME AS START
0466  BD 0002 R           JSR    NXTADR        TRY FOR PROM ADDRESS
0469  27 06               BEQ    :A3           NO.
046B  FE 000C R           LDX    ADR           YES.
046E  FF 000B R           STX    PROMAD
0471  CE 000B R  :A3      LDX    #PROMAD       VERIFY THAT RANGE <= 512
0474  0D                  SEC                  (FORCE BORROW)
0475  E6 07               LDAB   7,X           =ADDH+1
0477  E2 05               SBCB   5,X           =ADDL+1
0479  A6 06               LDAA   6,X
047B  A2 04               SBCA   4,X
047D  81 02               CMPA   #2            SHOULD BE 1 OR 0
047F  2C 08               BGE    :A4           TOO BIG.
0481  EB 01               ADDB   1,X           ALSO SHOULD NOT OVERSTEP PROM
0483  A9 00               ADCA   0,X
0485  A8 00               EORA   0,X
0487  84 FE               ANDA   #H'FE
0489  26 01               BNE    :A4           IT DOES.
048B  39                  RTS
048C  7E 0004 R  :A4      JMP    RNGERR        ADDRESS RANGE ERROR
                  *
                  *  TYPE RAM & ROM ADDRESS & DATA
```

```
                          *
     048F  CE 000D R   VERR     LDX    #ADDL        TYPE RAM ADDRESS
     0492                       CALL   P4HEX
     0492  3F             *     SWI
     0493     10          *     BYTE   P4HEX
     0494  FE 000D R            LDX    ADDL         NOW THE BYTE THERE
     0497  A6 00                LDAA   0,X
     0499  8D 95                BSR    P8BIN
     049B  B6 FBC6 A            LDAA   PROM+2+PIA   THEN PROM DATA
     049E  8D 90                BSR    P8BIN
     04A0  CE 0006 R            LDX    #PROMAD      NOW IF ADDRESS (LOW 8)
     04A3  A6 01                LDAA   1,X          DOES NOT MATCH RAM ADDRESS,
     04A5  A1 05                CMPA   5,X          =ADDL
     04A7  27 02                BEQ    :T
     04A9                       CALL   P4HEX        PRINT PROM ADDRESS
     04A9  3F             *     SWI
     04AA     10          *     BYTE   P4HEX
     04AB  BD 0006 R   :T       JSR    PCRLF
     04AE  86 40                LDAA   #64
     04B0  48                   ASLA                EXIT C=0, Z=0, V=1
     04B1  39                   RTS
                          *
                          *  PROM ADDRESS SETUP & DATA READ
                          *
     04B2  CE 0006 R   ADDRS    LDX    #PROMAD
     04B5  A6 01                LDAA   1,X          LOW 8 BITS
     04B7  B7 FBC4 A            STAA   PROM+PIA
     04BA  A6 00                LDAA   0,X          HIGH BIT
     04BC  CE FBC0 A            LDX    #PIA
     04BF  48                   ASLA                POSITION IT
     04C0  4C                   INCA                WITH DATA REGISTER SELECT
     04C1  48                   ASLA
     04C2  48                   ASLA
     04C3  A8 07                EORA   PROM+3,X     INSERT INTO CONTROL
     04C5  84 0C                ANDA   #12
     04C7  A8 07                EORA   PROM+3,X
     04C9  A7 07                STAA   PROM+3,X
     04CB  A6 06                LDAA   PROM+2,X     READ DATA
     04CD  39                   RTS
                          *
                          *  PROM VERIFY
                          *
     04CE  BD 0446 I   VFY      JSR    RASV         GO SETUP ADDRESSES
     04D1  8D 2D       :V       BSR    VFY1         VERIFY ONE LOCATION
     04D3  24 02                BCC    :N           NO ERROR, OR PRINTED.
     04D5  8D 3A                BSR    JVER         PRINT FIXABLE ERROR.
     04D7  8D 11       :N       BSR    INCAD        INCREMENT ADDRESSES
     04D9  20 F6                BRA    :V
                          *
                          *  PROM READ
                          *
     04DB  BD 0446 I   READ     JSR    RASV         SET UP POINTERS
     04DE  8D 02       :R       BSR    ADDRS        READ ONE BYTE
     04E0  FE 0000 R            LDX    ADDL
     04E3  BD 0008 R            JSR    SETMEM       STORE IN RAM
     04E6  8D 02                BSR    INCAD        NEXT!
     04E8  20 F4                BRA    :R
                          *
                          *  INCREMENT RAM/PROM ADDRESS POINTERS
                          *
     04EA  FE 0006 R   INCAD    LDX    PROMAD
     04ED  08                   INX
     04EE  FF 0006 R            STX    PROMAD
     04F1  FE 0000 R   INK      LDX    ADDL
     04F4  08                   INX
     04F5  FF 0000 R            STX    ADDL
     04F8  BC 000E R            CPX    ADDH
     04FB  26 B5                BNE    ADDRS
     04FD  7E 000A R   EXIT     JMP    MONITR       EXIT TO MONITOR
                          *
                          *  PROM DATA VERIFY, ONE BYTE
                          *
     0500  8D B0       VFY1     BSR    ADDRS        SET UP & READ A BYTE
     0502  FE 0000 R            LDX    ADDL         COMPARE TO RAM
     0505  A1 00                CMPA   0,X
     0507  27 07                BEQ    :X           OK: C=0, Z=1, V=0
     0509  43                   COMA                NO, IS IT FIXABLE?
     050A  AA 00                ORAA   0,X          I.E. NO RAM=0, PROM=1?
     050C  43                   COMA
     050D  26 02                BNE    JVER         YES.  C=1, Z=0, V=0
     050F  4C                   INCA
     0510  39          :X       RTS
     0511  7E 048F I   JVER     JMP    VERR         NO,  TYPE ERROR
                          *
                          *  PROM BURNER ROUTINE
                          *
     0514  BD 0446 I   BURN     JSR    RASV         SET UP PARAMETERS
     0517  7D 000F R            TST    COUNT        IF FULL, UNPARAMETERIZED,
     051A  26 1B                BNE    :B
     051C  7F 0006 R            CLR    PROMAD       DO BLANK CHECK
     051F  8D 91       :C       BSR    ADDRS
     0521  26 75                BNE    NOGOOD       AHA -- IT ISN'T
     0523  CE 0006 R            LDX    #PROMAD      INCREMENT PROM ADDRESS
     0526  6C 01                INC    1,X
     0528  26 F5                BNE    :C
     052A  6C 00                INC    0,X
     052C  A6 00                LDAA   0,X
     052E  46                   RORA
     052F  25 EE                BCS    :C
     0531  6F 00                CLR    0,X
     0533  20 02                BRA    :B
     0535  8D B3       :1       BSR    INCAD        ADVANCE TO NEXT
     0537  C6 03       :B       LDAB   #3           SET TRY COUNTER
     0539  8D C5                BSR    VFY1         CHECK THIS LOCATION
     053B  29 C0                BVS    EXIT         CAN'T PROGRAM 1 TO 0
     053D  FE 0000 R   :L       LDX    ADDL         GET DATUM
     0540  A6 00                LDAA   0,X
     0542  CE FBC0 A            LDX    #PIA
     0545  A7 06                STAA   PROM+2,X
     0547  A6 05                LDAA   PROM+1,X
     0549  84 F7                ANDA   #B'11110111  SET R/W TO W
     054B  A7 05                STAA   PROM+1,X
     054D  A6 07                LDAA   PROM+3,X     TURN IT AROUND
     054F  84 FB                ANDA   #B'11111011  (TO OUTPUTS)
     0551  A7 07                STAA   PROM+3,X
     0553  6F 06                CLR    PROM+2,X
     0555  63 06                COM    PROM+2,X
     0557  37                   PSHB
     0558  C6 14                LDAB   #20
     055A  8D 3F                BSR    MSEC         CLEAR TIMER
     055C  8D 3C       :P       BSR    MSEC         WAIT 1 MS BETWEEN PULSES
     055E  A6 01                LDAA   V50,X
     0560  84 F7                ANDA   #B'11110111  SET HIGH VOLTAGE
     0562  A7 01                STAA   V50,X
     0564  8D 35                BSR    MSEC         5 MS PULSE DURATION
     0566  8D 33                BSR    MSEC
     0568  8D 31                BSR    MSEC
     056A  8A 38                ORAA   #B'00111000  TURN OFF HIGH VOLTAGE
     056C  A7 01                STAA   V50,X
     056E  5A                   DECB
     056F  26 ED                BNE    :P           20 TIMES.
     0571  6F 06                CLR    PROM+2,X     CONVERT OUTPUTS TO INPUTS
     0573  A6 05                LDAA   PROM+1,X     TURN OFF WRITE
     0575  8A 30                ORAA   #B'00111000
     0577  A7 05                STAA   PROM+1,X
              10                IF     DELAY        OMIT IF NO POST PROGRAM DELAY
     0579  C6 0A                LDAB   #DELAY
     057B  8D 1E       :W       BSR    MSEC         DELAY (B) MS
     057D  5A                   DECB
     057E  26 FB                BNE    :W
                                IEND
     0580  33                   PULB
     0581  BD 0500 I            JSR    VFY1         CHECK IT:
     0584  29 0F                BVS    :J           BAD BIT SHOWED UP
     0586  27 AD                BEQ    :1           GOOD
     0588                       TYPE   21           NO, TYPE A NAK
              21          *     IF     21           0
     0588  86 15          *     LDAA   #21
                          *     IEND
     058A                 *     CALL   PRINTA
     058A  3F             *     SWI
     058B     11          *     BYTE   PRINTA
     058C  CE FBC0 A            LDX    #PIA
     058F  5A                   DECB                AND TRY AGAIN
     0590  26 AB                BNE    :L
     0592  BD 048F I            JSR    VERR
           0595 I      :J       EQU    *            GIVE UP
     0595  7E 0065 R   JBAD     JMP    PBADR        PRINT "BAD ADDRESS" AND QUIT
     0598  7E 0009 R   NOGOOD   JMP    ABORT
                          *
                          *  ONE MILLISECOND DELAY
                          *
     059B  6D 05       MSEC     TST    PROM+1,X     WAIT FOR CA1 TO FLOP
     059D  2A FC                BPL    MSEC
     059F  A1 04                CMPA   PROM,X       CLEAR IT (WITH A DATA READ)
     05A1  39                   RTS
                          *
                          *  MEMORY MOVE
                          *
               1                IF     MOVER
     05A2  BD 0001 R   MOVE     JSR    GETRNG       GET SOURCE ADDRESS RANGE
     05A5  BD 0002 R            JSR    NXTADR       GET DESTINATION STARTING ADDRESS
     05A8  27 EB                BEQ    JBAD         ERROR IF NONE
     05AA  FE 0000 R   :M       LDX    ADDL         GET BYTE
     05AD  A6 00                LDAA   0,X
     05AF  FE 000C R            LDX    ADR          STORE IT WITH VERIFY
     05B2  BD 0008 R            JSR    SETMEM
     05B5  08                   INX                 INCREMENT POINTERS
     05B6  FF 000C R            STX    ADR
     05B9  BD 04F1 I            JSR    INK          COMPARE TO END
     05BC  20 EC                BRA    :M           MORE
                                ELSE
                       MOVE     EQU    RAM
                                IEND
                          *
                          *  END OF MODULE
                          *
                                END

SYMBOL TABLE:

ABORT  0009 R   ADDH   000E R   ADDL   0000 R   ADDRS  04B2 I
ADR    000C R   BURN   0514 I   COUNT  000F R   DELAY  000A A
EXIT   04FD I   GETRNG 0001 R   INCAD  04EA I   INK    04F1 I
JBAD   0595 I   JVER   0511 I   MUNENT 0000 R*  MONITR 000A R
MOVE   05A2 I   MOVER  0001 A   MSEC   059B I   NOGOOD 0598 I
NXTADR 0002 R   P4HEX  0010 A   P8BIN  0030 I   PBADR  0065 R
PCRLF  0006 R   PIA    FBC0 A   PINIT  0416 I   PRINTA 0011 A
PROM   0004 A   PROMAD 0006 R   PSP    0443 I   PSPACE 0007 R
PXISTS 0003 R   RAM    FC00 A   RASV   0446 I   READ   04DB I
RNGERR 000A R   SETMEM 0008 R   V50    0001 A   VERR   048F I
VFY    04CE I   VFY1   0500 I

CHECKSUM = 93E8

LENGTH OF USECT =    0 (0000)
LENGTH OF ISECT =  424 (01A8)

       NO ERRORS,  NO WARNINGS,  THIS ASSEMBLY
```

Vectored from page 108

```
                                SYMBOL TABLE
* 01
A       0007      ABCH    0A34      ACPR    0BEF      ADDI    0DDF
ADD2    09DF      ADDZ    09D6      AORS    0BE1      APLS    0E55
B       0000      BBCH    0A3D      BCHK    0AF2      BCTL    0CB0
BMIN    0A22      C       0001      CCHK    0C44      CCMP    0B5B
CFCHE   0B4C      CHAR2   0F40      CCMI    0A16      COM2    0A28
COPT    00F4      COPY    0E02      CRIN    0940      CSIGN   0C69
CSTR    0C71      CTWO    0DE3      CVRT    0C9A      D       0002
DADD    0B06      DCLR    0B1D      DCMP    0BF4      DCOMP   0F3A
DECPT   0E86      DECR    0B04 *    DFXL    0ACE      DIGO    0D30
DIVIT   0DC6      DLST    0AE9      DRST    0AF8      DSUB    0B26
DTST2   0912      E       0003      ENT1    0C10      ENT2    0C0A
FPI     0F0F      EQU2    099C      EQUL    0994 *    ERR     0E1F
FINIT   0DDA      FINUP   0D8A      FLOAT   0AD5      FLTSG   0F15
FXL1    0AB2 *    FXL2    0AB3      GCHAR   0B41      GCHR    0DA0
GETA    0DA1      GETEX   0DC1      GETN    0F2A      GNUM    0ELC
GOON    093B      GOTV    0CE8      H       0004      INCR    0BB6
INCR2   0BC6      INCR3   0BC3      INDF1   0B84 *    INDFC   0B78
INEXP   0E9B      INP     0FFA      INPOP   0D25      INPUT   0E21
INT1    0E77      INTS    0E5F      INTR    0A7F      KPGO    0A70
L       0005      L003    0A46      L001    0A48      L002    0A1F
L003    0A4B      L128    0A4E      L129    0A51      L131    0A54
LADD    095C      LADS    0962      LASD    0A02      LDCP    0C39
LDIV    0900      LLTB    097F      LMCM    0A57      LMCP    0C40
LMUL    0A5F      LOOP    0DE5      LSFT    0D79      LSUB    0960
LXFR    0C24      M       0006      MADD    0AA4      MANT    09EF
MAXCH   003F      MCHK    0F67      MCHK1   0F68      MDGN    0D03
MDSKP   0D11      MINCH   00C0      MORO    0DA6      MSFH    0C79
MULT    0F70      MULTT   0051      N1      0F51      NCHK    0984
NNZRO   0CB2      NORM    0AA0      NORM1   0AAF *    NOTO    09AA
NZRO    0CCF      OFLW1   0876      OFLWC   0C58      OK1     0D0C
OUTR    0FFD      OVER    0C17      PLSV    009C      POPD    0D22
PSW     0006      PTVAL   0F47      READ    0008      REP3    0927
REPS    0C26      REP6    0AB9      SAVEN   0F19      SCALF   0E43
SCCFG   0B02      SCHAR   0ACB      SH10    098C      SHIFT   0F80
SIGN    0093      SP      0006      STCPC   0C53      SUBZ    09B9
TEN     0E1B      TENS    0F17      TLOOP   0D7C      TLP2    0D70
TOP     0F78      TRY1    0CF6      TRYP    0FE2      TRYTN   0E6F
TST1    0CFB      TST8    0C0B      TSTEX   0E36      TSTN    0FE7
TSTR    0BE7      TWOO    0DCC      UFLW1   0B68      UFLWC   0C62
USEIT   0EBD      WCHAP   0B9F      WIND    0B7F      WMANT   0B98
WOVR    0B71      WUND    0B63      WZER    0B8D      WZERC   0BAF
ZCHK    0ADA      ZMCHK   0ADA      ZROIT   0F30
```

VECTORED FROM PAGE 101

AGE, DEC. 1976, VOL.1,#13.

8080	AMS80	AMSAT 8080 STANDARD DEBUG MONITOR BY RICHARD C ALLEN & JOE KASSER - BYTE # 13, SEPT. 1976, VOL.2,#1. SUBMITTED BY JOE KASSER.	31-PTSL < 2 31-PTOD < 31-PACK *	10.00+0.60+2.00 3.00+0.18+1.75
6800	BAFCMP	BASIC ALGORITHMS FOR COMMON MATH FUNCTIONS BY MICHAEL P. BURTON - INTERFACE AGE, JAN. 1977, VOL.2,#1.	32-PTBL < 1 32-TEXT < 32-PACK *	3.00+0.24+1.00 1.00+0.06+1.00
8080	ECMSO	MICROCOMPUTER STOCK OPTIONS BY EDWARD CHRISTIANSON - INTERFACE AGE, FEB. 1977, VOL.2,#3.	33-PTBL < 0 33-HCBLF 33-HCBLF< 33-TEXT < 33-PACK *	6.00+0.36+2.00 2.00+0.12+1.25 INC. WITH PTBL
8080	BMRNG	RANDOM NUMBER GENERATOR BY BOB MARTIN - INTERFACE AGE, FEB. 1977, VOL.2,#3.	34-PTAL < 0 34-PTSL < 34-TEXT < 34-HCALF 34-HCALF<	6.00+0.36+2.00 5.00+0.30+1.75 1.00+0.06+1.00 1.00+0.06+1.00 INC. WITH PTAL
8080	RNDFGCST	RND FUNCTION GENERATOR CHI-SQUARE TEST PROGRAM BY BOB MARTIN - INTERFACE AGE, FEB. 1977, VOL.2,#3.	35-PTBL < 35-HCBLF< 35-PACK *	3.00+0.18+1.00 INC. WITH PTBL
8080	TTMOCSR	8080 MEMORY OBJECT CODE SEARCH ROUTINE BY T. E. TRAVIS - INTERFACE AGE, FEB. 1977, VOL.2,#3.	36-PTAL < 0 36-PTSL < 36-TEXT < 36-HCALF< 36-HCALF 36-PACK *	3.00+0.18+1.00 3.00+0.18+1.00 1.00+0.06+1.00 INC. WITH TEXT 1.00+0.06+1.00
8080	TDOMP	8080 OCTAL MONITOR PROGRAM BY THOMAS E. DOYLE - INTERFACE AGE, FEB. 1977, VOL.2,#3.	37-PTAL < 0 37-PTSL < 37-TEXT < 37-HCALF<	8.00+0.48+2.00 6.00+0.36+1.75 2.00+0.12+1.00 2.00+0.12+1.00

AN 8080 MEMORY OBJECT CODE SEARCH ROUTINE

By T. E. Travis

NEED FOR OBJECT CODE SEARCH ROUTINE

When attempting to modify software, it is often necessary to find and change every occurrence of a particular instruction sequence or every reference to some specified memory location. For example, if one wished to modify a teletype-oriented BASIC to use one of the available memory mapped video boards such as the Polymorphics VTI-1, it would be necessary to find all IN and OUT instructions directed toward the teletype port and change them to calls to input and output drivers for the video board. Since, in the case of vendor supplied software, a source listing of the program to be modified is often not available, a program to search through memory looking for and identifying all such calls is necessary.

SEARCH PROCESS

The program presented here requires no peripherals and will search memory starting at any desired location looking for a byte pattern of any arbitrary length. Every time the pattern is found, the program stops with the address of the first byte ***following*** the pattern in memory locations 26 and 27 (hex). Once those locations have been examined and noted, the program can be restarted and it will continue its search.

APPLICATION

As listed, the program begins at location zero, but it can be moved to any place in memory by changing all of the three byte instructions to point to their new targets. To use the program, load the starting address into memory locations 26 and 27, the length of the pattern into 28, and the pattern to be sought into 29 to 29 + n where n is the value stored in 28. The program is then initiated at location zero and will stop when it encounters the desired pattern. This will always happen because the program will eventually wrap around memory and compare the pattern with itself.

SEE MICROCOMPUTER SOFTWARE DEPOSITORY PROGRAM INDEX FOR COPIES OF THIS PROGRAM.

```
8080 MACRO ASSEMBLER, VER 2.3  ERRORS = 0 PAGE 1

                  ;
                  ;  PROGRAM NAME:  SERCH (8080 MEMORY SEARCH ROUTINE)
                  ;  PROGRAMMER:  T. E. TRAVIS
                  ;  STORAGE REQUIREMENTS:  41 BYTES PLUS PATTERN
                  ;  I/O REQUIREMENTS:  NONE
                  ;  REQUIRED SUPPORTING SOFTWARE:  NONE
                  ;
0000              SERCH   EQU   $
0000    2A2600            LHLD  ADDR          ;LOAD STARTING ADDRESS
0003              CONT    EQU   $
0003    112900            LXI   D,PATRN       ;LOAD PATTERN ADDRESS
0006    3A2800            LDA   LNGTH         ;LOAD PATTERN LENGTH
0009    47                MOV   B,A           ;MOVE LENGTH TO B
                  ;
                  ;  COMPARE PATTERN WITH MEMORY
                  ;
000A              COMP    EQU   $
000A    1A                LDAX  D             ;LOAD PATTERN BYTE
000B    BE                CMP   M             ;COMPARE WITH MEMORY
000C    C21800            JNZ   NOTEQ         ;BRANCH IF NOT EQUAL
000F    23                INX   H             ;INCREMENT MEMORY POINTER
0010    05                DCR   B             ;DECREMENT LENGTH COUNTER
0011    CA2200            JZ    EQUAL         ;BRANCH IF DONE
0014    13                INX   D             ;INCREMENT PATTERN POINTER
0015    C30A00            JMP   COMP          ;CONTINUE TESTING
                  ;
                  ;  PATTERN DID NOT COMPARE, ADVANCE MEMORY STARTING ADDRESS
                  ;
0018              NOTEQ   EQU   $
0018    2A2600            LHLD  ADDR          ;LOAD STARTING ADDRESS
001B    23                INX   H             ;INCREMENT
001C    222600            SHLD  ADDR          ;SAVE
001F    C30300            JMP   CONT          ;START TEST AGAIN
                  ;
                  ;  PATTERN DID COMPARE
                  ;
0022              EQUAL   EQU   $
0022    222600            SHLD  ADDR          ;SAVE HALT ADDRESS
0025    76                HLT                 ;HALT
                  ;
0002              ADDR:   DS    2             ;MEMORY START ADDRESS
0001              LNGTH:  DS    1             ;PATTERN LENGTH
0001              PATRN:  DS    1             ;TEST PATTERN
                  ;
                  ;  WHEN PATTERN FOUND, PROGRAM STOPS WITH ADDR POINTING TO
                  ;  FIRST BYTE FOLLOWING PATTERN
                  ;
                          END
NO PROGRAM ERRORS


8080 MACRO ASSEMBLER, VER 2.3  ERRORS = 0 PAGE 2

                         SYMBOL TABLE

 * 01

 A       0007      ADDR    0026      B       0000      C       0001
 COMP    000A      CONT    0003      D       0002      E       0003
 EQUAL   0022      H       0004      L       0005      LNGTH   0028
 M       0006      NOTEQ   0018      PATRN   0029      PSW     0006
 SERCH   0000 *    SP      0006
```

CIRCLE INQUIRY NO. 56

Letters to the Editor

Dear Editor:

I am a professional programmer and have been vaguely aware of the existence of the personal computer for about a year. Last week, while shopping, I passed by a computer store and stopped in to satisfy my curiosity. After looking at the system and reading some magazines, I left in a state of amazement and joy after discovering all of the hardware and software available to the home computerist. I'm very happy about finding a hobby that I can really "get into." For my first project, I would like to convert a B & W TV set into a video monitor. If any of the INTERFACE readers have any suggestions as to how to accomplish this task, I would be grateful to hear from them.

Brian O'Connell

Dear Editor:

I am writing this letter to inform the readers that the MITS-BASIC Software package will not run with the Z80 processor.

The problem has to do with the use of the parity flag. MITS seems to use it extensively.

I was wondering if any of your readers had some sort of a solution for this problem.

Joe T. Huffman

No sooner said than done — See the article A Z80 MITS 12K Extended BASIC Patches, by Martin D. Gray in this issue of INTERFACE AGE. This article provides required software patches to the MITS Extended BASIC for execution by the Z80 CPU.

Software Editor

Dear Editor:

. . . I am an inmate in the Washington State Penitentiary . . . Several of us here are trying to utilize our time by studying computers.

NAME WITHHELD

Dear Editor:

Thanks for publishing Dr. Wang's Palo Alto Tiny Basic in a more readable form than that which appeared in Dr. Dobb's Journal.

There is a glitch in the change sign routine which gives the "HOW" error message whenever a zero value or expression is preceded by a minus sign. For example: LET A = Ø; Let B = −A will not work.

To fix this change the CHGSGH routine at Ø486 by adding three instructions:

```
CHGSGN:
                MOV   A,H
added         { ORA   L
instructions  { RZ          ;zero value
              { MOV   A,H
                PUSH PSW
```

Arthur I. Larky PH. D

Dear Editor:

I have just started to read INTERFACE magazine and find it very difficult to put down. I really enjoy reading the articles that discuss the new versions of Star Trek that are becoming available and all the other articles that discuss various programs (Basic-diet-plan) and others. I am really amazed how you can just list the proper bits of information and just start plugging them right into the old computer banks. I am also glad that you put all your games and other lists in 8K Basic. I am going to be purchasing a IMSAI 8080 or an ALTAIR 8800b in the near future and through your magazine alone I will be able to start developing a library immediately.

There is a new topic I would really like to see discussed and that is the possibility and availability of voice recognition boards and apparatus for microprocessor like the IMSAI 8080 and the ALTAIR 8800b. I would really be fascinated and I would feel that investment into the microprocessors would be a good idea cause I could eventually expand it into a unit using voice recognition equipment as a peripheral.

Also is there a good book around that will teach me how to understand and use 4K BASIC all the way through extended BASIC (12K)? I would like a book that presents this information clearly and simply. I am looking forward to hearing from you very soon. I really hope to see an article about voice recognition and all its hardware come up soon. I just sent my subscription in to your magazine today so I won't miss any of the other exciting things you already put in your magazine. Thank you for your time.

Douglas Call

Dear Editor:

In that I am a beginner in computers and working with a small budget, I would like to see more how-to articles in your magazine. I cannot afford a language system right now, and since I'm starting from scratch with only a communications background, I am writing very simple iterations in machine language. I'm building an octal terminal for a little more speed in my programming. Your magazine seems to have a particular wealth of advertizements and a dearth of instruction and theory. Articles of some length on such things as instruction sets, and basic routines would be very welcome.

I am a technician in the Navy, and somewhat handicapped by a lack of literature overseas and the great cost of technical publications. I know that there are quite a few of us in the same situation. It is very difficult to learn microcomputers this way. I will be very happy to write you some articles when I learn something!

John Fitzpatrick

Dear Editor:

An article I would like to see in INTERFACE AGE would be about the use of hobbyist type equipment to solve some of the basic 'real world' problems of operating a robot in a house or shop type environment. Problems such as fail safe design, telemetery, navigation, and hazard avoidance.

Paul F. Grayson

MICRO-MARKET

FIFO FLEA MARKET

POWERTEC POWER SUPPLIES. New in sealed cartons. Three outputs 1 @ 5V-6A and 2 @ ±12-1.5A. Short and over voltage protected. Open, black aluminum frame. Perfect for μP and TTL work. $29.95. B. Adams, P.O. Box 146, Bryn Mawr, CA 92318.

IMSAI I-8080 kit — never assembled. Has 22 slot mother board, $490. PI04-1, $10;PIOMs, $16.50; SIO2-2, $119; RAM 4A-4, $105. Send SASE for info. First checks will take the gear. Include shipping costs; excess will be refunded, or arrange local pickup. S.R. Lewis, 1477 Barrington, #17, L.A. CA 90025.

SELL YOUR PRODUCTS IN FRANCE. American with a new French company wishes to represent U.S. products which have not as yet been exploited in Europe. If you feel your products are exportable - write to me with full details P. Studner, SUPER SEARCH, 39 Avenue de Breteuil, Paris 75007, France.

FOR SALE: Altair 8800, working, hardly used, ICs in sockets, flat-cable to front-panel, $440 or offer. BYTE Shop TV typewriter, new, working, $110 or offer. Dennis Feucht, 2015 S.E. 51st Ave., Portland, OR 97215. (503) 233-2648 or (503) 644-0161 ext. 6255.

FOR SALE: Two new unused MFE250B digital cassette drives - call best offer; Also have Remex papertape spooler $25 plus shipping; assembled morrow cassette board $100; Bug book III $10; Processor Tech 5KBASIC manuals and tape $10. Gary Alevy, 48D Clintwood Ct., Rochester, N.Y. 14620; (716)244-1133.

FOR SALE: Processor Technology 4K Static RAM assembled, works perfectly, meets all specs, with all documentation $150.00. Free unused papertape of 5K Basic with documentation included with RAM board. Barry Gerber, 720 Morse Ave., Placentia, CA 92670.

WANTED: SwTpc 6800 and TVT and KIM-1. Give details. Joe, 51625 Chestnut Road, Granger IN 46530.

FIFO FLEA MARKET ADS
★ PERSONALS FREE ★
Max. 10 lines per ad. Submit Ads to: FIFO Ads—**INTERFACE AGE** Magazine, P.O. Box 1234, Cerritos, CA 90701.

FOR SALE: ALTAIR 8800a, 16 slot m/brd, fan;2-4k dyn, 2-4k static, (16k total memory); ACR I/O; 3P+S I/O; TVT II w/64 chars/line; SCM Telewriter; software. $2150. Will Separate. Evenings - (312) 328-5610.

FOR SALE: 'Model 15 Page Printer' - Baudot teletype suitable OSI, MITS 680b computers. Excellent working condition, with current source and box of paper. Also, Friden 'Auxiliary Punch' — Baudot paper tape punch. Could be modified to ASCII with additional parts, not supplied. Probably works, not guaranteed; with stand and motorized tape takeup spool. Make offers for either to: G. A. R. Trollope, 433 Cherry Lane, Lewiston N. Y. 14092; (716)754-7222.

—WANTED: MITS ALTAIR disc (88-DCDD) with disc basic. Need not be in working condition, but must be complete. State age, condition, version basic, price and other particulars. Dana G. Peterson, 2129 Harper Street, Lawrence, KS 66044; (913) 841-2817.

FOR SALE: IBM 2311 compatible disc drive made by Marshall plus disc pack, $425. Teletype ASR 33 with auto-answer modem, auto-dial, stand. Three MITS 4K Dynamic boards, operating. Signetics 3000 Microprocessor Designer's Kit. Make offer. Lenny Heath, Bob Turnage, 86 Village Green, Greenville, N.C. 27834; (919)752-7813.

FOR SALE: ALTAIR 8800A completely assembled. Includes 18-slot motherboard. $550.00. K. Sisseon, 544 Wilkins, Apt. 1006, Detroit, MI 48201.

FOR SALE: ALTAIR 680b — assembled and running. Best offer over $300.00. Alan Dahlbom, 16 Forest Street, Cambridge, MA 02140.

FOR SALE: IMSAI, brand new, assembled and fully operational. with 22 card mother-board and 11 connectors. $850.00. Also memory boards. Sou Nam Kim, 10190 Vicksburg Drive, Cupertino, CA 95014. (408)996-0537...after 7:00 pm.

FOR SALE: Service Manual for Selectric I/O Writer. $28. R. Tierney, 4944 Cass St. #811, San Diego, CA 92109.

HELLO PARTY: I am interested in exchanging ideas/info. etc. relating to telephone applications for mini/micro computers, such as PABXs', Key Systems, etc. Call Norm...(213)653-3300.

FOR SALE: SPHERE 340 System 20K RAM, 1K system monitor, Full SIM board, Line printer w/tractor feed, Dual floppy, CRT, 4K EPROM board, 16K Mem. board less Mem. chips, and more. Lot's of software. $6,500. Wayne Smith, 227 S. State, Salt Lake City, UT 84111; (801)363-4941.

FOR SALE: 4K RAM boards for IMSAI or ALTAIR. High speed, low power (Music). Assembled and tested. $200.00 each. Dr. Michael Stimson, 1530 Grand Teton Ct., Chula Vista, CA, 92011.

MEMORY FOR SALE: GE Core Memory 16K by 40 Bits, all power and interface circuitry and documentation — been used with an 8080 CPU. Litton Industries Drum memory with full read/write electronics. Bryant MBM drum memory. By unit or best offer over $250.00 for lot. C. H. Eby, 7101 Mammoth Ave., Van Nuys, CA 91405; (213)988-1763.

FOR SALE: Poly-88 computer, new, never used. Assembled and tested, with the video tested by the factory. Includes 64 character line option and serial port. Power for expansion. Asking price $850.00. J. J. Goss, 5055 S.E. Roethe Road, Milwaukie, OR, 97222; (503)659-6601.

WANTED: Centronics (101, 102) Diablo, Datapoint (2200, 1100), or H.W., or H.P., equivalent. NCR, Entrex, Univac, tubes, printers, disk, diskette drives. Wango, Wang, Kennedy, Ampex, Pertec PEC tape drives, in any condition, working units, scrap frames or spare parts. Software Datapoint or H.W., H.P., equivalent. Pat Snyder, Box 31, Franklin Square, NY 11010.

FOR SALE: NEW Olivetti P-101 minicomputer. Easily programmed, permanent storage of programs on low cost magnetic cards, hard copy printout. Current list price $1,695. Will sacrifice for $995.00. Free delivery. Marc Seligman, 320 Bryn Mawr Ave., Bala Cynwyd, PA 19004.

HELP!! Where (besides GA) can I obtain a copy (p. tape/disk) of General Automation CAP 16 Assembler? Also need other data, programs, etc. Call Norm...(213)653-3300.

HP-45 CRYSTALS: I have some crystals to make the timebase in an HP-45 accurate so it can be used as a 10 event stopwatch. I am willing to sell these for $13 and will include some instructions on how to install it. Mark S. Egbert, 1514 S. 320 E. Orem, UT 84057. (801)224-2102.

PDP-8 MINI COMPUTER (Straight-8) for sale or trade. Mounted in attractive walnut cabinet (wife approved for den). Complete with maintenance manual, drawings, 4K Basic, 4K Focal, Editor, PAL III assembler, Disassembler, TTY interface, games, etc., etc. Perfect condition. Free DECUS membership available for additional software. Presently in use with ASR Model 33 teletype. Can be sold with or without teletype. Sale price w/o teletype $800.00. Rex Eagle, P.O. Box 10498, Charlotte, N.C. 28288. (704) 374-6557.

FOR SALE: Two MITS 4K static memory boards. Fully essembled and 100% operational. I am switching to 16K static boards. $160 each. David Milhouse, 2823 Griffa Ave., Columbus, IN 47201.

WANT SYSTEM, have R/C Gold Mine worth over $3300 to TRADE. Helicopters, planes, radios, cars, engines, field boxes and more. Interested in ALTAIR 8800B or IMSAI 8080 System in trade (will accept cash). If you're ready to move outdoors with your hobby please call A.B. Clark, (305)238-3408 nights, or (305)873-2528 days. Miami, FL.

SELECTRIC: One Dura Mach 10 wide platten Selectric based terminal with dual paper tape punches and readers. Computer interface available. IBM serviced and adjusted typer. Reconditioned and tested. Contact Ron Jenkins, 4490 Sirius Ave., Lompoc, CA. (805)733-2575 evenings except Mon. and Wed.

ADVERTISING INDEX

CAST YOUR VOTE

The authors in this issue would like to know your feelings as to which article you consider the best. You may cast your vote on each article or only one, which ever you prefer. The articles are graded on point scale 1 (lowest) to 10 (highest). At the end of the voting period all votes will be tallied and the highest scoring article will be announced and the author will receive $100 bonus.

NOTE: Use Original "Voting and Reader Service Card" only. Any other form is not acceptable.

ARTICLE	READER SERVICE VOTING NUMBER LOWEST		HIGHEST
8080 Octal Monitor Program *By Thomas Doyle*	151	—	160
Legion: An Experiment in Artificial Intelligence *By Marlin Ellers*	161	—	170
Microcomputer Stock Options *By Edward Christianson*	171	—	180
Warp Factors *By Tom Mazur*	181	—	190
Building A 12-Bit A/D Converter *By Roger Brown*	191	—	200
Computer Coupling To Teletype Loops *By B. D. Lichtenwalner*	201	—	210
The Black Box *By Ken Pugh*	211	—	220
Need A Better Random Number *By Robert Martin*	221	—	230
Basic Algorithms For Common Math Functions *By Michael Burton*	231	—	240

NEXT MONTH ANNOUNCEMENT OF DECEMBER'S BEST AUTHOR WINNER.

INTERFACE™

February 1977 Issue – Void after April 30, 1977

VOTING CARD

(Please type or print)

Name ______________________ Title ______________

Company ______________________ Phone (A/C) ______________

Address ______________________ ☐ Home ☐ Business

City ______________ State ______ Country ______ Zip ______

Please send information on items circled below.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210
211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330
331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390

My primary interests are in ☐ Hardware ☐ Software.
I am a ☐ Hobbyist ☐ Professional Engineer ☐ Businessman.

READER SERVICE CARD 2

INTERFACE™

February 1977 Issue – Void after April 30, 1977

(Please type or print)

Name ______________________ Title ______________

Company ______________________ Phone (A/C) ______________

Address ______________________ ☐ Home ☐ Business

City ______________ State ______ Country ______ Zip ______

Please send information on items circled below.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210
211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330
331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390

My primary interests are in ☐ Hardware ☐ Software.
I am a ☐ Hobbyist ☐ Professional Engineer ☐ Businessman.

READER SERVICE CARD

SUBSCRIPTION APPLICATION FORM

PAYMENT MUST ACCOMPANY FORM

SUBSCRIPTION FEE per year: $10.00 U.S., $12.00 Canada/Mexico
$18.00 Foreign.
Make check payable to: INTERFACE™ Magazine
P.O. Box 1234, Cerritos, CA 90701

Name ______________________ Title ______________

Company ______________________

Home Address ______________________

City ______________ State ______ Country ______ Zip ______

Date ______________ Signature ______________

READER SERVICE CARD 3

February 1977 Issue – Void after April 30, 1977

(Please type or print)

Name ______ Title ______
Company ______ Phone (A/C) ______
Address ______ ☐ Home ☐ Business
City ______ State ______ Country ______ Zip ______
Please send information on items circled below.

My primary interests are in ☐ Hardware ☐ Software.
I am a ☐ Hobbyist ☐ Professional Engineer ☐ Businessman.

SUBSCRIPTION APPLICATION FORM
PAYMENT MUST ACCOMPANY FORM
SUBSCRIPTION FEE per year: $10.00 U.S., $12.00 Canada/Mexico,
$18.00 Foreign.
Make check payable to: INTERFACE Magazine
P. O. Box 1234, CERRI'OS, CA 90701

Name ______ Title ______
Company ______
Home Address ______
City ______ State ______ Country ______ Zip ______
Date ______ Signature ______

FIRST CLASS
PERMIT No. 18
Cerritos, Ca. 90701

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Interface Hotline
P.O. Box 1234
Cerritos, Ca. 90701

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES
Interface Hotline
P.O. Box 1234
Cerritos, Ca. 90701
FIRST CLASS
PERMIT No. 18
Cerritos, Ca. 90701

FIRST CLASS
PERMIT No. 18
Cerritos, Ca. 90701
BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES
Interface Hotline
P.O. Box 1234
Cerritos, Ca. 90701

PLANNING TO MOVE?

Let us know 8 weeks in advance so that you won't miss a single issue of INTERFACE.

Attach old label where indicated and print new address in space provided. Also include your mailing label whenever you write concerning your subscription. It helps us serve you promptly.

If you have no label handy, print OLD address here.

Please Print

Name ______________________

Company ______________________

Address ______________________

City ______ State ______ Zip code ______

Affix Label ⬆ **Print NEW address here** ⬇

Name ______________________

Company ______________________

Address ______________________

City ______ State ______ Zip code ______

MAIL COMPLETED CARD TO:

INTERFACE™
P. O. Box 1234
Cerritos, California 90701

PLANNING TO MOVE?
Let us know 8 weeks in advance so that you won't miss a single issue of INTERFACE.
Attach old label where indicated and print new address in space provided. Also include your mailing label whenever you write concerning your subscription. It helps us serve you promptly.
If you have no label handy, print OLD address here.
Please Print
Name
Company
Address
City State Zip code
Affix Label Print NEW address here
Name
Company
Address
City State Zip code
MAIL COMPLETED CARD TO:
INTERFACE™
P. O. Box 1234
Cerritos, California 90701

Experience the excitement of owning the world's finest *personal* computer – IMSAI 8080

Waiting for you — all the incredible performance and power of the IMSAI 8080. And at a price you would normally pay for a fine home music system.

Introduced less than two years ago, the IMSAI 8080 is sold worldwide and acknowledged as the finest *personal* computer available.

WORLD OF USES

The IMSAI 8080 is a superbly engineered, quality computer. It is versatile, expandable and powerful, putting literally hundreds of applications and uses at your fingertips. Imagine sitting at your desk and enjoying interaction with your IMSAI 8080! Press the *on* switch and you're ready for game playing, research, education, business applications, or for laboratory instrument control. It has all the power you need, and more, to make your application ideas come alive.

GROWS WITH YOU

The IMSAI 8080 is designed for many years of pleasure. With its open-chassis engineering you can expand your system by adding peripherals and interfaces. The 22-slots and 28 amp power supply mean that you can plug-in today's, plus *tomorrow's* modules.

Right now you can add a module for displaying color graphics and characters on TV; a ready-to-use keyboard; small and large printers, and a single interface that lets you attach multiple devices including a cassette tape recorder. Expect the latest, exciting equipment from IMSAI. We are committed to leadership in this expanding technology.

EASY TO PROGRAM

With our BASIC language you can operate the IMSAI 8080 quickly and easily. Technically knowledgeable? Use our assembly language to develop sophisticated and unique software.

If you're thinking personal computer, treat yourself to the very best – IMSAI 8080

Send for free four-color brochure or $1 for catalog. Call or write for name of nearest dealer.

IMSAI

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093
TWX 910-366 7287

European Distributor
Harper Industry Products, Ltd.
6079 Sprendlingen 2
West Germany